MashZone NextGen User and Developer Guide

Version 9.10

April 2016

This document applies to MashZone NextGen Version 9.10 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2013-2016 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at http://softwareag.com/licenses.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at http://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at http://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: PR-UD-38-20150415

Table of Contents

Preface **43**
Additional Information and Support 43

Introducing Software AG MashZone NextGen **45**
Real-Time Intelligence 46
How Does MashZone NextGen Meet This Challenge? 47
MashZone NextGen Artifacts: A 'Black Box' View of the Flow of Information 49
MashZone NextGen Core Components 51
Tools and Users for MashZone NextGen Hub and the AppDepot 52
MashZone NextGen Hub 52
MashZone NextGen Enterprise AppDepot 54
MashZone NextGen Server 55
MashZone NextGen Repository 57
MashZone NextGen Analytics In-Memory Stores 57
MashZone NextGen APIs, Specifications and Extension Points 58
Enterprise Mashup Markup Language: The Mashup Domain Language 59
Real-Time Analytics Query Language 62
View API and Framework 63
MashZone NextGen Add-Ons 63
MashZone NextGen Real-Time Analytics Add-On with Terracotta BigMemory 63
MashZone NextGen Mobile 64
MashZone NextGen Add-On for SharePoint 64
MashZone NextGen Add-On for Excel 65
MashZone NextGen Add-On for Portals 66
MashZone NextGen concepts 66
What is ...? Why can't I? 68
Wires concepts 71
View concepts 73
Mashboard concepts 74

Working in MashZone NextGen Hub **77**
MashZone NextGen Hub Features 78
Your MashZone NextGen Hub Profile 82
Manage Your MashZone NextGen User Attributes 82
Manage Your Locale and Account Information 83
Reset Your Password 84
Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub 84
Common Tasks for Mashables, Mashups and Apps 85
Add Feedback 86
Add or Update Meta Data: Description, Category, Provider and Tags 87
Add or Update Custom Attributes 88

Add a Thumbnail....89
Change the Artifact Name....91
Change an Artifact's Locale....92
Change the Status....93
Feature or Unfeature....93
Add to Favorites or Bookmark....94
Delete....95
Review Dependencies....95
Share With Other Users....95
Export Results to CSV....96
Update Mashables, Mashups or Apps in Bulk....97
Find and Select Multiple Artifacts for Update....97
Find and Select Artifact Dependencies....98
Add To or Remove Multiple Artifacts From Favorites....99
Update the Category, Provider or Tags in Bulk....99
Change the Status of Multiple Artifacts....99
Delete Multiple Artifacts....100
Update Run Permissions for Multiple Artifacts....100
Update Caching for Multiple Mashables or Mashups....100
Feature or Unfeature Multiple Artifacts....101
Descriptions, Providers, Categories and Tags for Artifacts....101
Category....101
Description....102
Provider....102
Tags....102
Grant Permission to Run Mashables, Mashups and Apps....102
Grant Run Permissions for One Artifact....103
Find Users and Groups for Permissions....104
Choose MashZone NextGen Built-In Groups for Permissions....105
Grant Run Permissions to an Artifact's Dependencies....106
Grant Run Permissions to Published Apps in the AppDepot....107

Mashables....109
Types of Mashable Information Sources....110
Database Mashables....115
Database Mashable Support and Limitations....115
Database Mashable Names versus Database Names....116
Database Mashable Name Changes....116
Database Characters and Names That Are Not Supported....117
Default Operations Available for Database Mashables....118
Arbitrary SQL Queries for Database Mashables....124
Built-in Finders for Arbitrary Queries....124
Custom Queries....125
Database Mashables Responses....125
MashZone Feeds....128

REST Web Services....129
GET-Based REST Web Services....129
POST-Based REST Web Services....130
Snapshot Mashables....130
Spreadsheet Mashables....130
Valid Labels and Values for Spreadsheet/CSV Mashables....131
Syndicated Feeds (RSS/Atom)....131
Normalization and MashZone NextGen Support for RSS/Atom Formats....132
WSDL Web Services....133
WSDLs for SOAP Web Services....133
Security Policies and Configuration....133
WSDL Web Service Responses....134
Web Services Security (WSS) for WSDL Services....134
Built-in WSS Support for Policies and Policy Attachments....135
Built-in WSS Support for Policy Assertions and Tokens....136
Built-in WSS Support for Certificates and Encryption....136
Connect Information Sources as Mashables....136
Register Syndicated Feeds (RSS/Atom)....137
Register REST Web Services....138
REST Web Service URLs with Parameters....139
REST Web Services with Input Documents....140
Specify HTTP Request Headers....140
Register WSDL Web Services....141
Register Spreadsheets as Mashables....142
Register an Excel Worksheet File from MashZone NextGen Hub....143
Register an Excel Worksheet With a URL from MashZone NextGen Hub....144
Register XML Mashables....145
Register CSV Mashables....146
Register Basic Database Mashables....147
Register Custom Database Mashables....149
Enabling/Disabling Data Alteration Operations and Other Security Considerations....151
Viewing Details for Tables, Views or Stored Procedures....152
Removing Tables, Views or Stored Procedures from the Mashable....153
Removing Columns for Table or View Operations....153
Changing Column Names or Datatypes for Tables or Views....153
Removing Table or View Finders....154
Changing Finders for Tables or Views....154
Adding Finders for Tables or Views....155
Register SharePoint Lists....156
Register a Snapshot as a Mashable....158
Subscribe to Events from the Event Bus or Apama....158
Create or Edit MashZone Feeds....159
Edit an Existing MashZone Feed....159
Create a New MashZone Feed....160

Mashable Authentication with Security Profiles..... 161
HTTP Basic Authentication..... 161
CAS2 Authentication..... 162
NTLM Authentication..... 162
SSL Authentication..... 162
SSO Authentication..... 163
Custom Security Profiles..... 163
Using MashZone NextGen Global or User Attributes in Security Profiles..... 163
Configure Secure Connections for Mashables..... 164
Implement a Custom Security Profile Client..... 164
Register a Custom Security Profile..... 171
Configure HTTP Request Headers..... 174
REST Web Services..... 175
GET-Based REST Web Services..... 176
POST-Based REST Web Services..... 176
Running Mashables or Mashups and Other Tasks..... 176
Run and Preview Mashable/Mashup Data..... 177
Valid Date Formats for MashZone NextGen Mashables and Mashups..... 179
Update Mashable Endpoints..... 180
Update Mashable Security Profiles..... 181
Setting a Character Encoding for a Mashable..... 182
Take, View or Delete Snapshots..... 183
Take a Snapshot..... 183
Find and View Snapshots..... 183
View Snapshot Results in XML or JSON..... 185
Delete a Snapshot..... 185
Schedule Snapshots..... 185
Schedule Snapshot Jobs..... 185
Delete Snapshot Scheduled Jobs..... 186

Dashboards..... 187
Use dashboards in view mode..... 188
Display dashboards in MashZone NextGen..... 188
Use interactive filter in your dashboards..... 188
Use dynamic URL selection..... 189
Display dashboards without application header..... 190
Refresh data of dashboard components..... 191
Change column width and column sort order..... 191
Manage dashboards in edit mode..... 192
Create a dashboard..... 192
Edit a dashboard..... 192
Edit dashboard properties..... 193
Delete a dashboard..... 193
Manage dashboard permissions..... 194
Change dashboard style template..... 195

Create dashboard style templates... 195
Edit style templates... 196
Importing Dashboards... 197
Exporting Dashboards... 198
Configure dashboard views... 199
Add views in a dashboard... 199
Delete views in a dashboard... 199
Configure dashboard views... 200
Configure dashboard components... 200
Insert components in a dashboard... 200
Insert Line Charts in a dashboard... 201
Insert Column Charts in a dashboard... 202
Insert Bar Charts in a dashboard... 203
Insert Pie Charts in a dashboard... 203
Insert Table in a dashboard... 204
Edit Table display options... 205
Insert Speedometer Charts in a dashboard... 205
Insert Horizontal and Vertical Gauge Charts in a dashboard... 206
Insert Traffic Lights in a dashboard... 207
Insert Drop-down box in a dashboard... 208
Insert Input field in a dashboard... 209
Insert MashZone NextGen Apps in a dashboard... 210
Insert MashZone NextGen Mashups in a dashboard... 210
Insert Image in a dashboard... 211
Insert Label in a dashboard... 211
Assign data sources to dashboard components... 212
Assign data sources to Line chart... 213
Assign data sources to Column chart... 215
Assign data sources to Bar chart... 216
Assign data sources to Pie chart... 218
Assign data sources to Grid... 220
Configure thresholds of the Grid component... 222
Assign data sources to Speedometer chart... 224
Assign data sources to Horizontal and Vertical Gauge Chart... 226
Assign data sources to Traffic Light... 228
Assign data sources to MashZone NextGen App... 230
Assign data sources to MashZone NextGen Views... 230
Assign data sources to Drop-down box... 231
Assign data sources to Label... 232
Data source operators... 234
Assign Event data source... 234
Assign BigMemory data source... 235
Assign CSV data source... 236
Assign XML data source... 238
Assign ARIS Table data source... 240

Assign JDBC data source....241
Data transformation operators....243
Change data type....243
Behavior....243
Parameter....245
Error behavior....248
Examples....251
Filter rows....252
Behavior....252
Comparison operators....253
Error behavior....254
Conditional replace....254
Behavior....254
Parameter....254
Comparison operators....255
Error behavior....256
Arithmetic....257
Behavior....257
Parameter....257
Example....259
Move date....260
Behavior....260
Parameter....260
Error behavior....261
Example....262
Round date....262
Behavior....262
Parameter....263
Error behavior....264

Examples.... 264
Aggregate.... 266
Behavior.... 266
Parameter.... 266
Aggregation type.... 267
Error behavior.... 268
Examples.... 269
Insert column.... 270
Behavior.... 270
Parameter.... 270
Error behavior.... 271
Duplicate column.... 271
Behavior.... 271
Parameter.... 271
Delete column.... 272
Behavior.... 272
Parameter.... 272
Rename column.... 272
Behavior.... 272
Parameter.... 272
Error behavior.... 273
Create Input Parameters.... 273
Move components to front or back.... 274
Hide or display component header and border.... 275
Resize dashboard components.... 275
Copy and past components in dashboard.... 275
Define filters for dashboard components.... 276
Delete components from dashboard.... 277
Specify actions for dashboard components.... 277
Customize desktop.... 279
Set display size.... 280
Set grid lines.... 280

Mashups in MashZone NextGen Wires..... 281
Wires Features..... 282
Creating Mashups in Wires..... 284
Add a Mashup or Mashable as an Information Source..... 286
Add a Web Information Source with DirectInvoke..... 288
Add SharePoint Information Sources..... 290
Add a SharePoint List or List View to a Mashup..... 291
Add Results from a SharePoint Search or FAST Search to a Mashup..... 292
Add a Mashup or Mashable in a Loop..... 295
Preview Results..... 296
About Results in Wires..... 297
Stop Running a Block..... 297
Add Actions or Other Blocks..... 297
Aggregate Statistics..... 300
Add Input Fields Directly to Block Properties..... 303
Average Results..... 304
Count Results..... 304
Connecting Input Blocks to Block Properties in the Wires Canvas..... 306
Selecting Existing Input Blocks for Block Properties..... 306
Format Results as CSV..... 307
Decorate or Transform Data..... 307
Build Dates..... 309
Create a Document for Input..... 311
Add Fields or Structure Nodes..... 313
Map Other Results to the Output..... 313
Enter or Calculate Data..... 314
Adapt Results as a SharePoint External Content Type..... 314
Create an ECT-Enabled Mashup..... 315
Supported SharePoint Operations..... 315
Data Requirements for the Results of ECT-Enabled Mashups..... 316
ECT Block Properties..... 317
Updates to the Mashup's Results..... 317
Extract Partial Results..... 318
Filter Results..... 319
Filtering Techniques..... 322
Group Results..... 323
Set Conditions to Filter Items to Include in Groups..... 324
Add Calculations for the Group..... 326
Group Configuration Example..... 327
Add Input Parameters..... 327
Automatically Add Input Fields for Block Properties..... 328
Select an Existing Input Block for Block Properties..... 329
Manually Add and Connect Input Blocks to Block Properties..... 329
Join Results..... 330

Fine Tune the Joined Result....331
Upgrading Mashups with Join Blocks from Previous Versions....334
Map Results to Known Structures....334
Enter or Calculate Data....337
Merge Results....337
Properties for Web Feed Sources....337
Properties for Sources That Are Not Web Feeds....338
Merge SharePoint Lists....339
Strip Namespaces....340
Run Several Blocks Simultaneously....340
Pivot Data....341
Analyze Datasets with RAQL Queries....343
Dynamic RAQL Queries in Wires....344
Select Fields....345
Add an Item to a SharePoint List....346
Run a SQL Statement....348
Using Input Parameters as SQL Query Parameters....350
Build Strings....351
Sort Results....352
Transform Results....354
Drag Input Nodes to Add Nodes to the Output....355
Manually Add Fields or Structure Nodes....357
Map Input Nodes to the Output....358
Enter or Calculate Data....359
Built-in Functions for Decorator, Mapper and Transformer Blocks....359
Absolute....360
Add....360
Add to Date....361
Ceiling....361
Concatenate....361
Currency Value....361
Date Formatter....362
Date from Days....363
Date from Seconds....363
Distance Between Geo Points....363
Divide....363
Encode URI....364
Floor....364
Join With....364
Multiply....364
Not....365
Round....365
Set Decimal Places....365
String Matches....365
String Replace....365

Substring....366
Substring After....366
Substring Before....366
Subtract....367
Subtract from Date....367
To Boolean....367
To Camel-Case....368
To Date....368
To Datetime....368
To Decimal....368
To Integer....368
To Lower-case....368
To Number....369
To String....369
To Title-Case....369
To Upper-case....369
Build URLs....369
Copy and Update a URL....371
Provide the Complete URL Dynamically....371
Build a URL from Components....371
Select Fields or Paths for Block Properties with the Path Selector....372
Use MashZone NextGen Global or User Attributes in Block Properties....374
Changing Mashup Input Parameter Names....374
Valid Input Parameter Names....375
Add If/Else Conditions....375
Set Execution Conditions....376
Execution Condition Limitations....377
If/Else Examples....378
Add HTTP Headers to a Mashup....380
Advanced Properties: Controlling Blocks....381
Define Error Handling....382
Handle Empty Fields for WSDL Web Services....382
Use a Security Profile with DirectInvoke....382
Customizing the Selection XPath for Merge....383
Connect Mashup Blocks....383
Advanced Updates with the Output Block....384
Quickly Set All Inputs for the Mashup....385
Change the Mashup Result Character Encoding....385
Symbols to Escape in Block Properties....385
Date and Time Formats, Math and Sorting in Wires....386
Date/Times As Inputs to Mashables or Mashups....386
Converting Other Formats to Datetime....386
Date/Times For Date Math....386
Date/Times For Sorting....386
Manage Mashups in Wires....386

Quickly Find Mashables, Mashups or Other Blocks..... 387
Edit a Mashup..... 387
Save a Mashup..... 388
Turn Mashups On or Off..... 389
View Mashup or Mashable Details..... 389
View Custom Block EMML Code..... 390
Open the Mashup Artifact Page to Create Apps, Add Views or Take Snapshots..... 390
Edit Mashup XPath Expressions in Advanced Mode..... 391
Customizing Wires..... 392
Adding Custom Blocks to MashZone NextGen Wires Using Macros..... 392
Use Domains to Organize Custom Blocks..... 392
Define Inputs and Output for 'Wiring' the Macro..... 393
Loop Through a Well-Known Input..... 394
Make Macros Generic for Custom Blocks..... 396
Use or Construct XPath Expressions for Generic Macros..... 397
Configure Properties for Custom Blocks..... 403
Update Labels and Icons for Custom Blocks in Wires..... 406
Block Labels..... 406
Property Labels..... 407
Block Icons..... 407
Add Tooltips and Help in Custom Blocks..... 409
Tooltips..... 409
Add a Help Button and Help Topic..... 410
Make Custom Block Properties Required..... 411
Configure Field and Line Sizes for Custom Block Properties..... 412
Configure Strings, Numbers, Dates or Boolean Properties in Custom Blocks..... 415
Passwords..... 416
URLs..... 417
Integer Numbers..... 417
Decimal Numbers..... 418
Boolean Choices..... 419
Pick and Format Dates or Times..... 419
Configure Dynamic Path Choices and Results for Custom Blocks..... 423
Limit Dynamic Choices and Forbid Literal Entries..... 424
Get the Chosen Path as Text..... 426
Limit Custom Block User Entries to a List of Values..... 429
Literal List Values..... 429
Dynamic List Values from a Macro Input Parameter..... 432
Dynamic List Values from a Known Source..... 433
Advanced Custom Block and Property Configuration..... 435
Additional Configuration for Blocks..... 436
Additional Configuration for Block Property Fields..... 437

Mashups in EMML..... 441
Writing Mashups in EMML..... 442

Creating, Testing, Saving and Publishing Mashups....443
Creating a Mashup Script with EMML....443
<mashup>....449
EMML Namespaces....450
<operation>....451
Declaring Mashup and Macro Variables and Parameters....452
Valid Names for Variables and Parameters....452
Datatypes for Variables and Parameters....453
Default and Constructed Values....454
Variable and Parameter Scopes....455
Referencing Parameters and Variables....457
Working Samples....457
<input>....457
<output>....459
Commonly Supported Output Character Encodings....461
<variables>....462
<variable>....463
Declaring Namespaces....465
Declaring Data Sources....467
Named Datasources versus the Default Datasource....468
Explicitly Defining the Connection....469
Dynamic Connections....470
Working Samples....470
<datasource>....470
Configuring Datasource Drivers....472
Invoking Component Mashups and Mashables....473
<invoke>....473
<invoke> Examples....475
<directinvoke>....478
securityprofile....483
param....483
directinvoke Examples....484
Controlling Component Mashup/Mashable Invocation at Runtime....491
Issuing SQL Statements Directly to a Datasource....491
<sql>....492
<sql> Examples>....494
<sqlUpdate>....498
<sqlUpdate> Examples....499
<sqlcall>....501
Transforming Intermediate Results....502
Using XPath Functions to Transform Data....503
<assign>....503
<assign> Examples....505
<filter>....506
<filter> Examples....507

<group>.... 510
<group> Examples.... 511
<sort>....517
<sort> Examples.... 518
<annotate>....520
<annotate> Examples.... 521
<script>....523
Adding User-Defined Scripting Code to Mashups.... 524
Scripting Languages....524
Import Script Libraries....525
Include Scripts Directly.... 525
Access to Mashup Variables....526
Access to Java Classes....526
Working Samples.... 527
Adding User-Defined or Third-Party Java Classes for JavaScript Access..... 527
Installing and Configuring JRuby for Mashups.... 528
Deploying Groovy or JavaScript Scripts in MashZone NextGen....528
<xslt>.... 529
Known Limitations.... 530
Deploy the Stylesheet and Related Resources.... 530
Custom XPath Functions and other XSLT Extensions....531
Combining Component Mashable or Mashup Results....531
<join>....531
<select>.... 532
<join> Examples....533
<merge>.... 535
<merge> Examples.... 537
Constructing the Mashup Result, Inputs or Intermediate Variables.... 539
Literal XML Structures with Literal or Dynamic Data.... 539
<constructor>....540
<constructor> Examples....540
<appendresult>....541
<appendresult> Examples....542
<select>.... 543
<columns>.... 544
<column>....544
<select> Example....544
Controlling Mashup Processing Flow....545
<if>....546
<elseif>....546
<else>....547
<if> Examples....547
<for>.... 548
<for> Example....549
<foreach>....549

<fuse>....551
<foreach> Examples....552
<while>....555
<while> Example....555
<break>....556
<break> Example and Working Samples....556
Concurrent Mashup Processing Flows....556
<parallel>....557

<sequence>....557

Working Samples....557

Database Mashable Transactions....557
<presto:beginTransaction>....558
<presto:commitTransaction>....559
<presto:rollbackTransaction>....559
SQL Transactions....560
<sqlBeginTransaction>....561
<sqlCommit>....561
<sqlRollback>....562
Handling or Throwing Exceptions....562
<try>....563
<catch>....564
Examples....565
<throw>....572
<return>....573
Lightweight Error Handling for Component Mashups and Mashables....574
Supporting Debugging....575
<display>....575
<display> Examples....576
<assert>....577
<assert> Examples....579
Adding Metadata to Mashups....581
<emml-meta>....583
<presto:macro-meta>....583
<block>....584
<label>....584
<help>....585
<icon>....586
<uiconfig>....587
<parameters>....588
<parameter>....588
<required>....588
<type>....588
<datetimefield>....590

<list>....590
<values>....591
<option>....591
<xpath>....592
<presto:presto-meta>....593
<user-meta>....594
Migrating Mashups to a New EMML Version....594
Migrating Mashups to Use a New Feature or Namespace....595
Expressions for Mashups....596
A Brief Introduction to XPath 2.0....596
XML Escaping in URLs and Expressions....600
Defining Custom XPath Functions....601
Write a Custom XPath Function Class....601
Add the Function to the Classpath for the MashZone NextGen Server....602
Add the Function to a Mashup Script or Macro....602
Using XQuery Expressions in Construction Statements....603
Dynamic Mashup Expressions....604
Dynamic Expressions in Literal XML....605
Dynamic Expressions in XPath Expressions....605
Dynamic XPath or Other Syntax in Mashups....606
Advanced Mashup Techniques....606
Adding HTTP Headers to the Mashup Result....607
Adding Authentication Headers for Component Mashables....608
Construct Mashable Authentication Headers....608
Invoke the Mashable with Authentication Headers....608
Normalizing Data for Joins, Grouping or Filtering....609
Use the Custom XPath Function in Your Mashup or Macro....610
Converting and Working with Dates and Times....610
Using the ISODateFormatExt Custom XPath Function....611
ISODateFormatExt(dateText, fromFormat, toFormat)....612
Converting Numbers in Floating Point Notation....613
Adding Special Characters to Data....613
XML Character Entities....613
Using MashZone NextGen Attributes in Mashups....614
MashZone NextGen Global and User Attributes....615
MashZone NextGen Session Attributes....615
Using MashZone NextGen Attributes in Mashup Statements....616
Removing Duplicates With Filtering....617
XPath Axes....618
Setting the Mashup Response Character Encoding....619
Setting the HTTP Content Type Header in the Response....619
Setting the Character Encoding in the Response XML Declaration....619
Wrapping Results or Variables in CDATA Sections....619
Web Clipping with a Mashup Script....621
Handling HTML Responses....623

Handling JSON Responses or Inputs....624
Convert JSON to XML....625
Post JSON to a Web Service Using <directinvoke>....625
Accept JSON as Input and Modify with <script>....626
Wrapping POJO Classes with Mashups....626
Using Java Classes in a <script> Statement....627
Wrapping WSDL Web Services That Return HTML Results....628
Pre-Processing/Post-Processing Mashables with Mashups....630
Pre-processing Mashup....630
Post-processing Mashup....631
Self-Joins with a Single Dataset....633
Using XQuery for Outer Joins....634
Sample Mashable Results....634
Constructing the Join....636
Defining and Using Custom Mashup Statements with Macros....637
<macro>....638
<macro> Examples....639
Creating and Registering Macros....643
Create an Individual Reusable Macro....643
Create a Reusable Macro Library....645
<macros>....645
Including Macro Domains in Mashup Scripts or Macros....646
<include>....646
Calling a Macro....647
Macro Reference Namespace....647
Macro Domains....648
Passing Input Parameters to the Macro....648
Receiving Macro Results....649
Calling Macros From Another Macro....649
<macro:custom-macro-name>....650
Making Mashup Scripts Dynamic....650
Dynamic Mashup Syntax....651
Use Parameters/Variables in Attributes....651
Define XPath or Other Syntax Using <template>....651
Escape XML Delimiters Inside <template> Expressions....652
Use Dynamic Mashup Expressions Inside XPath Expressions....652
Create Generic Mashup Scripts....652
<template>....653
Generating Mashup Scripts Dynamically....654
<macro:externalDynamicEMML>....654
Using the Built-in Mashup Query Web Service....655
User Metadata Queries....656
MashZone NextGen Metadata Queries....657
Mashable Dependency Queries....658
Mashup Samples....659

Mashup Utilities.... 664
Testing Mashup Scripts from the Command Line.... 665
Publishing Mashup Scripts from a Command Line.... 666
Updating Mashup Scripts from the Command Line.... 667
Publishing and Managing Macros from the Command Line.... 668
Profiling Mashup Performance.... 669
Configuring Profiling Logging.... 670
Mashup Profile Logs.... 670
EMML Reference.... 670
Declarations Group.... 676
Macroincludes Group.... 677
Statements Group.... 677
Variables Group.... 678

Working in the Mashup Editor.... 679
Editing Short Cuts and Tips.... 680
Automatically Generating <invoke> Statements in Mashup Editor.... 681
Adding Macro Calls Automatically in the Mashup Editor.... 681
Editing a Mashup in the Mashup Editor.... 682
Test Mashup Scripts or Macros in Mashup Editor.... 683
Publish a Mashup Script in the Mashup Editor.... 684

MashZone NextGen Analytics.... 685
What is MashZone NextGen Analytics?.... 686
Getting Started with MashZone NextGen Analytics.... 689
A Basic RAQL Query.... 689
Structure, Format and Access to Datasets with RAQL.... 691
The RAQL Query Syntax.... 692
Use Plain Functions to Update, Select or Sort Rows.... 693
Datatype Information for Loaded Datasets.... 695
The Stream/Document Boundary.... 695
Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics.... 696
Group and Analyze Rows.... 699
Group and Analyze Rows with Row Detail.... 701
Where to Go Next.... 704
RAQL Queries.... 704
Techniques to Load Data.... 704
Select Techniques.... 705
Over Techniques.... 705
From Techniques.... 706
Where Techniques.... 707
Group By Techniques.... 707
Having Techniques.... 708
Order By Techniques.... 708
Multi-DataSet Operations.... 708

Escape Characters for RAQL Queries....710
Load Data with <sql>....710
Add a Datasource to MashZone NextGen to Connect to the Database....711
Stream the Database Results and Query with RAQL....711
Query and Store to the MashZone NextGen Analytics In-Memory Stores....712
Load Data with <invoke>....712
Load Snapshot Data with <snapshot> or <raql>....714
Snapshot Queries....715
Load Snapshots in a Named Variable....715
Load Snapshots Anonymously....716
Choose Snapshots by a Time Period....717
Choose Snapshots for a Scheduled Job....717
Select Distinct Values....718
Alias Names for Columns or Calculations....719
Use Alias Names in Other Clauses....719
Use Alias Names to Handle HTML Column Content....720
Plain Functions on Select Columns....720
Dataset Paths, Names and Datatypes....721
Default rules for schema detection from XML....721
Adding Paths to Clarify RAQL Row Detection....722
Providing Dataset Path and Datatype Information in a Schema....722
Valid Names for Datasets, Columns, Aliases, Paths and Functions....727
RAQL Reserved Keywords....729
From Subqueries....731
Limit the Rows Returned....732
Where Complex Conditions....733
Multiple Required Conditions....733
Combining Logical Conditions....733
Comparing Dates or Numbers....734
Calculations in Conditions....735
Negative Comparisons....735
Where Text Like Patterns....736
Where in Sets....737
Where Between a Range....738
Parameters in Where Clauses....739
Literal Values in Conditions or Expressions....739
Plain Functions in Where....740
Sort Directions and Multi-Level Sorts....741
Plain Functions in Sorts....742
Multi-Level Group Calculations....743
Having Clause to Filter Groups....754
Partitions and Windows....754
Running Aggregates....757
Number or Rank Rows....758
Analytic Functions for Partitions and Windows....759

Joins and Other Multiple-Dataset Operations....760
Join: Merge Columns for Dataset Rows....761
Union: Append Like Datasets....761
Intersect: Find Common Dataset Members....762
Except or Minus: Find All Less the Intersection....763
Creating Dynamic RAQL Queries....764
Migrate RAQL Queries from Version 3.7 to 3.8....765
RAQL Datatypes and Data Formats....771
Supported Data Formats for RAQL....771
Valid RAQL Datatypes....772
RAQL Operators....773
Built-In RAQL Functions....775
Built-In Plain Functions....775
Built-In Analytic Functions: Aggregate and Window....785
Create and Add User-Defined Functions for RAQL Queries....804
Set Up Your Development Environment....805
Write Plain Functions for RAQL....806
Configure, Compile, Deploy and Test User-Defined Functions....807
Write Window Analytic Functions for RAQL....811
Write Aggregate Analytic Functions for RAQL....817
Use a Factory for Function Overloading....823
Statistics and Analytics Third-Party Libraries....823
RAQL Query Syntax Reference....823
Working with MashZone NextGen Analytics In-Memory Stores....857
Declared Versus Dynamic In-Memory Stores....857
External versus Internal Datasets....859
About BigMemory and the MashZone NextGen Analytics In-Memory Stores....861
Initial Memory Configuration and Performance....861
In-Memory Dataset Management....861
Memory Management and Configuration....861
BigMemory Configuration, Cache Sizing and Pinning....863
Store Data in MashZone NextGen Analytics In-Memory Stores....865
Store Data in Declared In-Memory Stores....865
Set Unique Keys for Dataset Rows....866
Append Query Results Repeatedly....867
Clear Existing Data from In-Memory Stores....871
Manage Memory for Dynamic In-Memory Stores....871
Load Data from the MashZone NextGen Analytics In-Memory Stores....872
RAQL Extension to EMML Statements for Mashups....876
<loadfrom>....877
<raql>....879
<snapshot>....880
<storeto>....882
<dropcache>....884
RAQL Extensions to EMML Statements for Streaming....885

Data Access as Documents with No Streaming....885
Data Access as Streaming Datasets....885
Stream Partitions....886
RAQL Extensions to <variable> for File Data Sources....886
RAQL Extensions for Dataset Schemas....887
RAQL Samples....887

Views for Mashups and Mashables....895
Built-in MashZone NextGen Views....901
Add Views to Mashables and Mashups....901
About Desktop and Mobile View Compatibility....903
Manage Views....904
Common Chart Techniques....904
Change Series Labels....904
Change the Series Plot to Column, Line or Area....905
Add a Target Line....906
Include Dynamic Content in Captions and Labels....906
Dial Gauge....908
Characteristics....909
Configure This View....909
Add a Gauge to This View....910
Configure a Dial Gauge....910
Area Chart....912
Characteristics....912
Configure This View....912
Bar Chart....914
Characteristics....915
Configure This View....915
Bubble Chart....916
Characteristics....917
Configure This View....917
Bulb Gauge....919
Characteristics....919
Configure This View....919
Add a Gauge to This View....920
Configure the Bulb Gauge....920
Bullet Gauge....921
Characteristics....922
Configure This View....922
Add a Gauge to This View....922
Configure a Bullet Gauge....923
Candlestick Chart....924
Characteristics....924
Configure This View....925
Column Chart....926

Characteristics....926
Configure This View....927
Cylinder Gauge....928
Characteristics....928
Configure This View....929
Add a Gauge to This View....929
Configure the Cylinder Gauge....929
Doughnut Chart....930
Characteristics....931
Configure This View....932
Feed Reader....933
Characteristics....933
Funnel Chart....934
Characteristics....934
Configure This View....935
Gantt Chart....936
Characteristics....936
Configure This View....937
Visual Effects of Optional Data Configuration for Gantt Charts....938
Gauges....941
Geo Map....941
Characteristics....942
Configure This View....942
ISO 3166-1 Alpha 2 Country Codes....943
Google Map....944
Characteristics....944
Configure This View....944
Grid and KPIs View....946
Characteristics....946
Configure This View....947
Move or Remove Columns....947
Add Columns and Assign Values....948
Change Column Headings, Width and Text Formatting....948
Show As: Change the Column Format for Different Types Data....949
Number Formatter....951
Spark Charts, Bullet Graphs and Percentages....951
Use Templates to Combine Data, Add Literal Text or Add HTML Tags....958
Add Key Performance Indicators (KPI)....959
Precedence for Column Settings and KPI Rules....966
Intensity Map....966
Characteristics....966
Configure This View....967
ISO 3166-1 Alpha 2 Country Codes....967
Kagi Chart....968
Characteristics....969

Configure This View....969
LED Gauge....970
Characteristics....970
Configure This View....971
Add a Gauge to This View....971
Configure the LED Gauge....971
Line Chart....972
Characteristics....973
Configure This View....973
Real-Time Line Chart....974
Characteristics....975
Configure a Real-Time Line Chart....975
Linear Gauge....980
Characteristics....980
Configure This View....981
Add a Gauge to This View....981
Configure the Linear Gauge....981
Marimekko Chart....982
Characteristics....983
Configure This View....983
Pareto Chart....984
Characteristics....985
Configure This View....985
Pie Chart....986
Characteristics....986
Configure This View....987
Record Details View....988
Characteristics....988
Configure This View....989
Pyramid Chart....989
Characteristics....990
Configure This View....991
Radar Chart....992
Characteristics....992
Configure This View....993
Scatter Chart....994
Characteristics....994
Configure This View....994
Sparkline....995
Characteristics....996
Configure This View....996
Sparkline Column....997
Characteristics....998
Configure This View....998
Sparkline win_loss....999

Characteristics....1000
Configure This View....1000
Spline Area Chart....1001
Characteristics....1001
Configure This View....1001
Spline Line Chart....1002
Characteristics....1003
Configure This View....1003
Step Line....1004
Characteristics....1004
Configure This View....1005
Template View....1006
Create a Custom View....1006
Map Result Fields to the Template....1008
Map Results Fields That Contain HTML Markup....1009
Iterate Through Repeating Results....1011
Add Conditional Logic....1012
Use CSS in Template Code....1013
Use Images in Template Code....1015
Use JavaScript in Template Code....1015
Examples....1017
Thermometer Gauge....1020
Characteristics....1021
Configure This View....1021
Add a Gauge to This View....1022
Configure the Thermometer Gauge....1022
Zoom Line Chart....1023
Characteristics....1024
Configure This View....1024
Advanced Configuration Properties....1025
Background....1025
Canvas....1027
Text Formatting....1028
Layout....1031
Labels....1032
Advanced....1036
More Properties....1045
Creating Pluggable Views or Libraries....1045
Views and Libraries in MashZone NextGen....1045
Implement and Add a Pluggable Library or View....1047
Implement a Pluggable View Class....1048
A Hello World Pluggable View Class....1048
Pluggable View Classes: Using Mashable/Mashup Data....1054
Selecting Columns for View Configuration....1056
Formatting and Rendering Response Data....1057

Pluggable View Classes: Triggering and Handling Events..... 1059
Register and Trigger View-Specific Events in Pluggable Views.....1059
Support DataTable Row Selection Events in Pluggable Views..... 1061
Example: Complete D3 Pie with View-Specific and DataTable Events..... 1063
CSS Styles, Help and Thumbnails for Pluggable Views..... 1066
Configuration Properties for Pluggable Views or Libraries.....1067
Import Pluggable Views and Libraries.....1073
Library Projects Folders and Configuration Files..... 1073
Ant Tasks and Sample Build File to Import Pluggable Views or Libraries..... 1074
Import Pluggable Library/View Examples.....1076
Managing Updates and Library Versions..... 1085
Library Versions and Dependencies..... 1085
Handle Minor Library Updates..... 1087
Handle New Library Releases or Major Updates.....1090

Apps and Workspaces..... 1091
Create a Basic App..... 1093
Configure App Input Parameters.....1094
Select and Arrange Views.....1096
Preview and Finalize the Presentation.....1096
Save the App.....1100
Publishing, Managing, Sharing and Using Apps.....1101
Embed an App in HTML Pages..... 1102
Publish an App in SharePoint..... 1102
Edit App Properties.....1104
Input Parameters..... 1104
Layouts..... 1106
Themes..... 1107
Pagination and Miscellaneous Properties..... 1108
Sorting and Filtering Properties.....1109
Device Compatibility and General Properties..... 1110
Edit Properties for Custom or Workspace Apps..... 1110
About Desktop and Mobile Compatibility for Apps.....1111
Title Bars and Toolbar Buttons in Apps.....1112
Buttons in the Default Title Bars..... 1113
Publish Apps to the AppDepot..... 1114
Create Workspace Apps with Mashboard.....1116
Working in Mashboard..... 1121
Find Views, Apps or Workspaces in Mashboard..... 1123
Finding Views..... 1123
Views Added Directly to a Workspace.....1124
Basic App Wrapper for Views..... 1124
Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace.....1125
Edit Workspaces.....1126

Move, Cut, Copy, Paste or Remove Apps or Views in Workspaces..... 1127
Copy a Workspace..... 1128
Control Title Bars and Borders..... 1128
Set Input Parameters, Pagination, Isolation or Auto-refresh..... 1130
Set Data Share or Sort and Filter Tools for Views..... 1131
Grant Run Permissions to a Workspace..... 1131
Add or Delete Rows and Columns to a Layout..... 1131
Adding Columns or Rows..... 1132
Deleting Columns or Rows..... 1132
Increasing the Maximum Number of Columns or Rows..... 1133
Set Size, Spanning and Padding in Column or Table Layouts..... 1133
Initial Widths and Heights for Column or Table Layouts..... 1133
Control Padding for Column or Table Layouts..... 1134
Resize Apps in Column or Desktop Layouts..... 1134
Change Cell Width or Height with Spanning in Table Layouts..... 1137
Fixed Row Heights for Table Layouts..... 1138
Add, Delete, Arrange or Configure Tabs or Pages in Layouts..... 1138
Add Tabs or Pages..... 1140
Delete Tabs or Pages..... 1140
Change Tab or Page Names..... 1140
Change the Order of Pages..... 1140
Set Other Tab or Page Configuration..... 1140
Set Desktop Layout Colors and Properties..... 1141
Web Page Layout for Tabs or Pages..... 1142
Add a Real-Time Line Chart..... 1142
Configure the Real-Time Line Chart..... 1142
Add a PPM Chart to a Workspace..... 1147
Additional PPM Chart Options..... 1149
Update a PPM Chart..... 1149
Update the PPM Chart Favorite..... 1149
Update PPM Chart Selection Properties..... 1150
Control User Jumps to PPM..... 1150
Wire Selection and Filter Events for PPM Charts..... 1150
Set Filter Properties for Wiring PPM Charts..... 1151
Add a Dimension or Key Indicator Criteria..... 1153
Add Additional Criteria for an Existing Dimension or Key Indicator..... 1154
Filter Wiring Example for a PPM Chart App..... 1155
About PPM Filters..... 1158
Add a Drill Down App in a Workspace..... 1158
Add a Filter for Apps in a Workspace..... 1161
Edit or Delete Filter Controls..... 1163
Add an Input Form for All Apps In a Workspace..... 1164
Edit a Workspace Form..... 1166
Workspace Form Fields..... 1168
Title Block..... 1168

Input Block....1169
Text Area Block....1169
Select Block....1169
Radio Block....1169
Add Google Gadgets to a Workspace....1170
Add an HTML Widget to a Workspace....1171
Add Other Media as Objects to a Workspace....1172
Add Web Pages to a Workspace....1173
Add Gadgets With JavaScript to a Workspace....1174
Wiring App Interactions in a Workspace....1175
Events and Topics for Wiring Basic and Custom Apps....1178
Publish Topic Messages for Events in Basic Apps....1179
Subscription Topics for Basic Apps....1179
Hiding Input Forms for Apps in a Workspace....1181
Synchronizing Wired Apps When a Workspace Loads....1181
Update Input Parameters on Workspace Load....1182
Synchronize Row Selection on Workspace Load....1184
Manage App Interaction Wiring in Workspaces....1185
Toggle Wiring Subscriptions Off or On....1186
Edit Subscription Property Mappings....1186
Working in the MashZone NextGen Enterprise AppDepot....1186
AppDepot Tabs and Toolbars....1187
Managing Pending Apps in the AppDepot....1187
About MashZone NextGen Mobile....1188
Installing MashZone NextGen Mobile Apps in Mobile Devices....1189
Login and Get Started in MashZone NextGen Mobile....1189
Working in MashZone NextGen Mobile Apps....1189
MashZone NextGen Mobile App Features....1190

Custom Apps....1191
Working in the App Editor....1194
Find and Open an App....1195
Edit and Test an Open App....1196
Download an App....1197
Update an App....1198
Customize a Basic App or View....1198
Customize Themes for Basic Apps and Views....1200
CSS Selectors for Themes....1206
Customize the Layout for Basic Apps....1207
Including Built-in Tools in a Custom Layout....1214
Layout Roles....1217
Customize Built-In Chart Attributes....1218
Create Fully Custom Apps in the App Editor....1225
Create Custom Apps from the Base App Package....1226
Declare, Get and Set Properties in Custom Apps....1227

Get Properties or Property Values.... 1228
Set Property Values.... 1229
App Dimensions and Resizing.... 1236
Set App Dimensions.... 1238
Support Resizing in Custom Apps.... 1238
Handle Resize Events.... 1238
Enable User-Initiated or Automatic Refreshes.... 1243
User-Initiated App Refreshes.... 1243
Automatic App Refreshes.... 1246
Handle Exceptions.... 1247
Wiring App Interactions.... 1249
Tightly-Coupled and Loosely-Coupled Interactions.... 1252
Tightly Coupled Interaction for Custom Apps.... 1253
Publisher App for Tightly Coupled Interactions.... 1254
Subscriber App for Tightly Coupled Interactions.... 1257
Embedding Tightly Coupled Apps.... 1260
Wildcard Subscriptions for Tightly-Coupled Interactions.... 1262
Enable Loosely-Coupled Interactions in Custom Apps.... 1264
Publisher App for Loosely-Coupled Interactions.... 1264
Subscriber App for Loosely-Coupled Interactions.... 1272
Basic App Support for Loosely-Coupled Interactions.... 1282
The Completed Sample Apps.... 1283
Declare App Topics and Payloads.... 1285
Registering Sample Mashables or Mashups.... 1287
Register Samples Mashables.... 1287
Register Sample Mashups.... 1288
Override Browser Caches for Updates to App Resources.... 1289
The Structure of a MashZone NextGen App.... 1290
Parameters for App URLs.... 1294
App Specification Reference.... 1298
<app>.... 1299
<authors> or <author>.... 1302
<authors>.... 1302
<author>.... 1302
<dependson>.... 1303
<resource>.... 1303
<description>.... 1303
<help>.... 1304
<icons> or <icon>.... 1305
<icons>.... 1305
<icon>.... 1305
<presto-meta>.... 1306
App Layout.... 1306
App Themes.... 1307
Charts Theme.... 1307

Device Compatibility Flags....1307
Toolbar Button Flags....1308
View Configuration....1308
<properties> or <property>....1310
<properties>....1310
<property>....1310
<content>....1313
<requires> or <require>....1313
<requires>....1313
<require>....1314
<title>....1316
<topics> or <topic>....1316
<topics>....1316
<topic>....1316
App Packages and App Files....1318

MashZone NextGen Add-Ons....1321
The MashZone NextGen Add-On for Excel....1322
Install the MashZone NextGen Toolbar in Excel....1323
Before You Start....1323
Installation....1323
MashZone NextGen Toolbar for Excel....1324
The MashZone NextGen Toolbar Add-On Configuration File....1325
Configuring the Connection to MashZone NextGen....1325
Work with MashZone NextGen Mashup and Mashable Results in Excel....1328
Import MashZone NextGen Mashup or Mashable Results....1328
Selecting the Result Nodes to Import....1331
Change Parameters for an Imported Mashup or Mashable....1332
Changing Imported Mashup or Mashable Data....1333
Refresh MashZone NextGen Mashup or Mashable Results....1333
Remove an Imported Mashup or Mashable....1333
Export Excel Worksheets or CSV Files as Mashables in MashZone NextGen....1334
Register a Worksheet as a Spreadsheet Mashable in MashZone NextGen....1334
Valid Spreadsheet Content....1334
Manual or Automatic Updates to Spreadsheet Content....1334
Register a Range of Cells From the Current Worksheet....1335
Register a Worksheet From an Excel File in Excel....1336
Register a CSV File as a Mashable....1338
Valid Spreadsheet Mashables....1339
Update Mashables Manually from Worksheets....1339
Remove an Excel Mashable from MashZone NextGen....1341
The MashZone NextGen Add-On for Portals....1341
Installing the MashZone NextGen Add-On for Portals....1342
Download the MashZone NextGen Add-On for Portals....1342
Download and Install MashZone NextGen....1342

Add the MashZone NextGen Add-On for Portals to Your Portal..... 1343
Update Configuration for the MashZone NextGen Add-On for Portals..... 1343
Login and Get Started.....1344
MashZone NextGen Add-On for Portals Architecture..... 1345
Migrating to New Versions of the MashZone NextGen Add-On for Portals.....1347
Download the New MashZone NextGen Add-On for Portals Release..... 1347
Upgrade MashZone NextGen to the Matching Release..... 1348
Backup Portal Configuration.....1348
Install the New MashZone NextGen Add-On for Portals Release.....1348
Clear the Portlet Cache in the Portal.....1348
Adding MashZone NextGen Mashups to Portal Pages.....1349
Adding MashZone NextGen Apps to Portal Pages.....1351
Managing and Updating Mashups or Apps in Portal Pages..... 1353
Portlet Preferences.....1354
Change the Display Size for a Mashup or App..... 1354
Change Mashup or App Input Parameters..... 1354
Enable the App Forms for Input Parameters, Sorting and Filtering.....1355
Show the App Title Bar.....1355
Change the Portlet Title..... 1357
Update the Mashup Layout for Views.....1357
Update Mashup or App Pagination Settings.....1357
Update App Theme..... 1358
Update App Portlet Performance Preferences.....1358
MashZone NextGen Add-On for SharePoint (P4S).....1358
An Introduction to the MashZone NextGen Add-On for SharePoint.....1360
Maximize the Potential of Information.....1360
Empower SharePoint Users with Powerful Visual Tools..... 1360
Maximize SharePoint ROI.....1361
MashZone NextGen Add-On for SharePoint in Action.....1362
MashZone NextGen Add-On for SharePoint Features..... 1362
Getting Started..... 1365
Installing MashZone NextGen Add-On for SharePoint..... 1366
Installing the MashZone NextGen Add-On for SharePoint Solution.....1367
Before Your Start.....1367
Install the P4S Solution.....1368
What is Installed.....1369
Troubleshooting Installation for the P4S Solution..... 1369
Upgrading to New Releases of P4SP.....1371
Install or Upgrade MashZone NextGen Servers for P4S..... 1372
P4S Configuration and Administration, SharePoint 2010.....1372
Planning Authentication for P4S 2010..... 1372
User Authentication Experiences for P4S 2010.....1373
Authentication Designs for P4S 2010..... 1373

SSO + Token Authentication.....1375

SSO + Cookie Authentication..... 1377

No SSO, Manual Authentication..... 1377

Configuring MashZone NextGen Add-On for SharePoint 2010..... 1378
Activate P4S 2010 Features..... 1378

Enabling P4S Features in Farms..... 1378

Enabling P4S Web Parts for Site Collections..... 1379

Manually Enabling P4S Features for a Site..... 1379

Configure Authentication for P4S 2010..... 1379

Configure the Secure Store Service..... 1380

Create Secure Store Target Applications..... 1380

Create or Manage SSO Token Server Names..... 1383

Disable Cookie Forwarding for MashZone NextGen Server Overrides..... 1384

Configure MashZone NextGen Server Connections in SharePoint 2010..... 1385

Add a MashZone NextGen Server Connection to SharePoint..... 1386

Edit a MashZone NextGen Server Connection..... 1389

Duplicate Connections to MashZone NextGen Servers..... 1390

Complete Miscellaneous Configuration for P4S 2010..... 1390

Set the Scope of Apps in the App Gallery..... 1391

Set the Default Sandbox Mode for apps..... 1391

P4S Farm and Site Administration, SharePoint 2010..... 1392
P4S Configuration and Administration, SharePoint 2007..... 1392
Planning Authentication for P4S 2007..... 1392
User Experiences for P4S 2007..... 1393
Authentication Designs for P4S 2007..... 1393

SSO + Token Authentication..... 1394

SSO + Cookie Authentication..... 1396

No SSO, Manual Authentication..... 1396

Configuring MashZone NextGen Add-On for SharePoint 2007..... 1396
Activate P4S Features for SharePoint 2007..... 1397

Enable P4S Features in Farms..... 1397

Enable P4S Web Parts for Site Collections..... 1398

Activate Custom P4S Actions for Site Collections..... 1398

Activate the MashZone NextGen Administration Feature for Site Collections. 1399

Configure Authentication for P4S 2007..... 1400

Configure the SSO Service in Windows..... 1401

Configure the SSO Service in SharePoint..... 1402

Create SSO Applications for P4S..... 1403

Configure SSO Token Server Names for P4S..... 1404

Disable Cookie Forwarding for MashZone NextGen Server Overrides for Sites..... 1405

Configure MashZone NextGen Server Connections in SharePoint 2007..... 1406

Add a MashZone NextGen Server Connection to SharePoint 2007..... 1406

Edit a MashZone NextGen Server Connection..... 1409

Duplicate Connections to MashZone NextGen Servers..... 1409

Complete Miscellaneous Configuration for P4S 2007..... 1410

Set the Scope of Apps in the App Gallery..... 1410

Set the Default Sandbox Mode for Apps..... 1411

Farm and Site Administration for P4S 2007..... 1411

SharePoint Credentials for Mashables and Mashups..... 1411

Add or Manage SharePoint Credentials in Your MashZone NextGen Hub Profile. 1413

Add a Global Attribute for Agent-Based SSO Authentication with SharePoint..... 1414

MashZone NextGen Add-On for SharePoint User Guide..... 1414

Publishing Apps and Mashups to Web Parts..... 1416

Other Tasks..... 1416

Create a Web Part Page..... 1416

Add a Mashup Web Part to a Web Part Page..... 1419

Publish a Mashup to a Mashup Web Part..... 1421

Mashup Web Part Toolbar..... 1423

Update Mashup Web Part Properties..... 1424

Change the Web Part Title..... 1425

Change the Mashup Server Connection for a Mashup Web Part..... 1426

Change the Mashup ID..... 1426

Publish Mashup Results as Native SharePoint Lists..... 1427

Publish Mashup Results as External SharePoint Lists..... 1428

Requirements and Limitations..... 1429

The Mashup-to-External-List Process..... 1430

Create an External Content Type for a Mashup..... 1430

Create an External Content List from a Mashup..... 1433

Troubleshooting URL Loading Problems..... 1433
Register a SharePoint List as a MashZone NextGen Mashable..... 1434
Update Mashup or App Input Parameters..... 1434
Add an App Web Part to the Web Part Page..... 1436
Publish an App to an App Web Part..... 1438
App Web Part Toolbar..... 1441
Update App Web Part Properties..... 1441
Change the Web Part Title..... 1443
Hide or Show App or View Titles and Toolbars..... 1443
Change Pagination or Themes..... 1444
Enable Anonymous Access..... 1444
Change the Mashup Server Connection for an App Web Part..... 1444
Change the App ID..... 1445
MashZone NextGen Add-On for SharePoint User Guide..... 1445
Publishing Apps and Mashups to Web Parts..... 1447
Other Tasks..... 1447
Create a Web Part Page..... 1447
Add a Mashup Web Part to a Web Part Page..... 1450
Publish a Mashup to a Mashup Web Part..... 1452
Mashup Web Part Toolbar..... 1454
Update Mashup Web Part Properties..... 1455
Change the Web Part Title..... 1456
Change the Mashup Server Connection for a Mashup Web Part..... 1457
Change the Mashup ID..... 1457
Publish Mashup Results as Native SharePoint Lists..... 1458
Publish Mashup Results as External SharePoint Lists..... 1460
Requirements and Limitations..... 1460
The Mashup-to-External-List Process..... 1461
Create an External Content Type for a Mashup..... 1462
Create an External Content List from a Mashup..... 1464
Troubleshooting URL Loading Problems..... 1465
Register a SharePoint List as a MashZone NextGen Mashable..... 1465
Update Mashup or App Input Parameters..... 1466
Add an App Web Part to the Web Part Page..... 1468
Publish an App to an App Web Part..... 1470
App Web Part Toolbar..... 1472
Update App Web Part Properties..... 1472
Change the Web Part Title..... 1474
Hide or Show App or View Titles and Toolbars..... 1474
Change Pagination or Themes..... 1475
Enable Anonymous Access..... 1475
Change the Mashup Server Connection for an App Web Part..... 1475
Change the App ID..... 1476

MashZone NextGen Development and APIs..... 1477

Features of the MashZone NextGen REST or MashZone NextGen Connect APIs....1479
Working with theMashZone NextGenREST API....1480
Get the Mashable/Mashup Specification....1481
Login or Set Guess Access....1481
Build and Send the Request....1482
Cross Domain Requests....1483
Session Management....1484
Use MashZone NextGen Connect for Javascript....1484
Use PC4JS in Custom Apps....1484
Use PC4JS in Other Web Applications or Web Sites....1486
Guest or Authenticated Access with PC4JS....1488
Use Mashable/Mashup Technical Specs....1489
Paginate Mashable or Mashup Responses....1491
MashZone NextGen Headers Used in Pagination....1492
Pagination Using the MashZone NextGen DataTable API....1492
Map MashZone NextGen Attributes to Artifact Input Parameters....1502
MashZone NextGen Attribute Mapping Expressions....1503
Define MashZone NextGen Session Attributes....1505
MashZone NextGen APIs....1507
MashZone NextGen REST APIs and MashZone NextGen Connect....1507
Pluggable Views, Custom Apps and Common JavaScript Client Libraries....1508
Miscellaneous Extension APIs....1508
MashZone NextGen REST API....1509
Handle Authentication....1509
Invoke Mashables or Mashups....1510
MashZone NextGen Headers/Parameters....1511
MashZone NextGen Request Headers/Parameters....1511
MashZone NextGen Response Headers....1516
Obsolete JUMP Headers....1517
MashZone NextGen Snapshot API....1517
Take or Delete Snapshot Methods....1518
Take a Snapshot Immediately....1518
Take a Temporary Snapshot and Save Separately....1519
Delete Snapshots....1520
Get Snapshot Methods....1520
Get Snapshot Collections By Criteria....1520
Get One Snapshot By ID....1523
Schedule Snapshot Job Methods....1523
Manage Scheduled Snapshot Job Methods....1526
Find Jobs....1526
Suspend Jobs....1526
Resume Jobs....1527
Delete Jobs....1527
About DataTable Events....1527
DataTable Events for Views and Apps....1527

MashZone NextGen Platform API Console....1529
MashZone NextGen Platform Services....1530
The JackBe Universal Message Protocol (JUMP)....1531
JUMP Requests....1532
JUMP Responses....1532
Input Parameters for JUMP Requests....1534
Parameter Values....1535
Adding Headers for Mashables to JUMP Requests....1535
Namespaces and Namespace Prefixes....1536
MashZone NextGen APIs....1536
MashZone NextGen REST APIs and MashZone NextGen Connect....1536
Pluggable Views, Custom Apps and Common JavaScript Client Libraries....1536
Miscellaneous Extension APIs....1537
MashZone NextGen REST API....1538
Handle Authentication....1538
Invoke Mashables or Mashups....1539
MashZone NextGen Headers/Parameters....1539
MashZone NextGen Request Headers/Parameters....1540
MashZone NextGen Response Headers....1545
Obsolete JUMP Headers....1546
MashZone NextGen Snapshot API....1546
Take or Delete Snapshot Methods....1547
Take a Snapshot Immediately....1547
Take a Temporary Snapshot and Save Separately....1547
Delete Snapshots....1548
Get Snapshot Methods....1548
Get Snapshot Collections By Criteria....1549
Get One Snapshot By ID....1551
Schedule Snapshot Job Methods....1552
Manage Scheduled Snapshot Job Methods....1555
Find Jobs....1555
Suspend Jobs....1555
Resume Jobs....1555
Delete Jobs....1556
About DataTable Events....1556
DataTable Events for Views and Apps....1556
MashZone NextGen Platform API Console....1558
MashZone NextGen Platform Services....1559

MashZone NextGen Administration....1561
Getting Started with the MashZone NextGen Server....1563
Additional MashZone NextGen System and Software Requirements....1564
Additional Recommendations for MashZone NextGen....1564
Recommendations for MashZone NextGen Add-On for SharePoint (P4S)....1565
Recommendations for the MashZone NextGen Add-On for Excel....1566

Recommendations for the MashZone NextGen Add-On for Portals..... 1566
Requirements for Optional Views.....1566
MashZone NextGen Browser and Mobile Device Compatibility.....1567
MashZone NextGen and MashZone Database Compatibility..... 1567
What is Installed with MashZone NextGen.....1569
MashZone NextGen Installation Folders.....1570
Start and Stop the MashZone NextGen Server..... 1570
Start the MashZone NextGen Server.....1570
Startup Account for the MashZone NextGen Server With MashZone NextGen Add-On for SharePoint.....1571
Stop the MashZone NextGen Server.....1572
Startup Considerations..... 1572
Manage Licenses for MashZone NextGen, Universal Messaging and BigMemory..... 1572
Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution..... 1574
Troubleshooting Connections to the MashZone NextGen Repository.....1574
Move MashZone NextGen and MashZone Repositories to Microsoft SQL Server.1575
Move the MashZone NextGen and MashZone Repositories to MySQL.....1579
Move the MashZone NextGen and MashZone Repositories to Oracle..... 1582
Move the MashZone NextGen and MashZone Repositories to PostGres.....1586
Integrate Your LDAP Directory with MashZone NextGen..... 1590
Defining LDAP Connection Configuration..... 1592
Defining the Authentication Scheme..... 1592
Defining the Authorization Scheme.....1593
Enabling MashZone NextGen Application Queries for All LDAP Users or Groups for Permissions..... 1595
Use the Default MashZone NextGen User Repository..... 1597
Manage Users..... 1597
Create Users..... 1598
Edit, Grant Permissions and other User Management Tasks..... 1598
Manage User Groups.....1599
Automatically Assign New Users to Groups..... 1600
What's Next..... 1601
MashZone NextGen Security..... 1602
Change technical user password..... 1603
Authentication and Guest Access..... 1604
User Authentication..... 1605
Valid Credentials.....1605
Sessions and Timeouts.....1606
Enabling Guest Access..... 1606
Default User Accounts.....1607
Authentication with Single Sign-On Solutions..... 1607
Configuration for Agent-Based SSO Solutions..... 1608
Configuration for the CAS SSO Solution..... 1611
Implementing a Custom SSO Filter..... 1614

Authentication with Digital Certificates/SSL... 1614
Configure the MashZone NextGen REST API to Use Certificate Authentication... 1616
Configure Alternate User ID Extraction... 1617
Configure Dynamic User Support... 1617
Configure Additional Certificate Validation... 1618
Authorization Policies and Permissions... 1619
Grant User Access to MashZone NextGen with Built-in Groups... 1621
Built-In MashZone NextGen User Groups and Permissions... 1622
Access Policies Using MashZone NextGen Built-In Groups... 1622
Artifact Permissions for Users with Run Permissions... 1623
Automatically Grant Run Permissions to Users and Groups... 1624
Set View Permissions with a Search Filter... 1625
Enable or Disable Authorization... 1625
MashZone NextGen Server Configuration... 1626
Memory Configuration for the MashZone NextGen Server... 1627
Configuration When MashZone NextGen Uses Only Heap Memory... 1627
Configuration When MashZone NextGen Uses Heap and Off-Heap Memory... 1629
Support International Character Sets and Locales... 1631
MashZone NextGen Repository Encoding and Timezone... 1632
Mashable, Mashup and App Encodings and Locales... 1633
Date, Time and Numeric Display Options... 1633
Message Log and Default Locales... 1634
Change the MashZone NextGen HubTheme... 1634
Set the default chart theme... 1637
Edit style templates... 1637
Configure the MashZone NextGen Server with Custom Ports... 1638
Change MashZone NextGen Server Ports... 1638
Change MashZone NextGen Repository Ports... 1639
TomEE Application Server Port... 1639
Configure the MashZone NextGen Server to Work with a Proxy Server... 1640
Define a Proxy Server Whitelist for MashZone NextGen... 1641
Using Regular Expressions in a Whitelist... 1641
Configure MashZone NextGen for SSL and Digital Certificates... 1643
The Certificate Store and Certificates... 1645
Configure Mutual SSL Between Users and MashZone NextGen... 1646
Configure Mutual SSL Between MashZone NextGen and Mashable Information Sources... 1646
One-Way SSL to MashZone NextGen... 1647
One-Way SSL to Mashable Information Sources... 1647
One-Way SSL to Information Sources Using <directinvoke> in Mashups... 1648
Configure HTTPS and Certficate Stores in the Application Server... 1648
Configure Certificate Stores in MashZone NextGen... 1649
Update SSL Configuration for Java... 1649
MashZone NextGen Logging... 1650
Configure Logging for the MashZone NextGen Server... 1650

Turn Audit Logging On or Off..... 1652
MashZone NextGen Notifications..... 1653
Configuring a Mail Server for MashZone NextGen..... 1654
Update the User Email Attribute from LDAP..... 1655
Configure MashZone NextGen Connections to SharePoint..... 1655
Connection Patterns Between MashZone NextGen Servers and SharePoint..... 1656
Add SharePoint Connections to the MashZone NextGen Server..... 1659
BigMemory for Caching, Connections and MashZone NextGen Analytics..... 1661
Caching for the MashZone NextGen Server..... 1662
Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores..... 1664
Declare BigMemory Stores for MashZone NextGen Analytics..... 1667
Declare a New In-Memory Store..... 1667
Modify a Declared In-Memory Store..... 1670
View Details for Declared In-Memory Stores..... 1670
Manage Dynamic BigMemory Stores for MashZone NextGen Analytics..... 1671
Add an External Dynamic In-Memory Store Connection..... 1671
Delete External Dynamic In-Memory Store Connections..... 1672
Configuring Mashable/Mashup Response Caching..... 1673
Enable and Configure Response Caching..... 1673
Cache Responses by Default and Disable Exceptions..... 1674
Cache Responses for Exceptions Only..... 1675
Controlling Response Cache Entries Dynamically..... 1676
Response Caching Example..... 1677
Manage Data Sources and Drivers..... 1679
Add a Data Source..... 1679
Edit, Test or Remove Data Sources..... 1681
Add or Manage JDBC Drivers..... 1681
Upgrade JDBC connections..... 1682
Upgrade JDBC connections of MashZone versions 9.5 to 9.9..... 1682
Upgrade JDBC connections of Presto version 3.9 and 9.9..... 1683
Upgrade JDBC configuration of Presto version 3.9 and 9.9..... 1683
Configure the Default Operations Generated for Database Mashable..... 1684
Manage Categories for Mashups, Mashables and Apps..... 1686
Manage Providers for Mashups, Mashables and Apps..... 1686
Work With MashZone NextGen Attributes..... 1687
Manage MashZone NextGen Global Attributes..... 1688
Expose User Attributes from the User Repository in MashZone NextGen..... 1689
Manage Artifact Attributes..... 1690
Add an Artifact Attribute Definition..... 1690
Edit or Delete Artifact Attribute Definitions..... 1690
Disable Mashup Features..... 1691
Configure HTTP Response Header Forwarding..... 1692
Configure Mashable HTTP Request Timeouts..... 1693
Enable or Disable the Snapshot Feature..... 1693

Set Web Feed Normalization..... 1694
Handle SOAP Encoding Errors for WSDL Services..... 1694
Add XML Schemas to the Wires Mapper Block..... 1695
Integrated MashZone Server Configuration and Administration..... 1696
Basic Settings for the Integrated MashZone Server..... 1696
Tune Memory/Caching for the Integrated MashZone Server..... 1697
Tune MashZone Memory and Cache Configuration Manually..... 1698
Automatically Tune MashZone Memory and Cache Configuration..... 1699
Create New MashZone Resource Directories..... 1700
Manage MashZone Resource Directories..... 1700
Configure Proxy Information for the Integrated MashZone Server..... 1701
Enable the Geocoding Operator for MashZone Feeds..... 1701
Import or Export MashZone Feeds..... 1701
Import Existing MashZone Feeds..... 1701
Export MashZone Feeds from MashZone NextGen..... 1703
Enable the MashZone Feed Editor..... 1703
Enable the MashZone Administration..... 1704
Event Service Configuration and Administration..... 1704
Manage EDA Event Sources..... 1705
Identify the Event Type Store directory..... 1706
Create EDA Event Sources..... 1706
Edit EDA Event Sources..... 1712
Duplicate EDA Event Sources..... 1717
Delete EDA Event Sources..... 1717
Import EDA Event Sources..... 1718
Export EDA Event Sources..... 1718
Manage Apama Event Sources..... 1719
Create Apama Event Sources..... 1719
Edit Apama Event Sources..... 1724
Duplicate Apama Event Sources..... 1729
Delete Apama Event Sources..... 1730
Import Apama Event Sources..... 1730
Export Apama Event Sources..... 1731
Start or Stop an Event Source..... 1731
Restart all Event Source..... 1731
Process Performance Manager (PPM) Integration..... 1732
Manage PPM Connections..... 1732
Create PPM Connections..... 1733
Edit PPM Connections..... 1734
Delete PPM Connections..... 1735
Import PPM Connections..... 1735
Export PPM Connections..... 1736
Import the PPM Chart App..... 1736
MashZone NextGen and MashZone Repositories..... 1737
Tuning the MashZone NextGen or MashZone Repository Connection Pool..... 1738

Synchronize the MashZone NextGen Repository and MashZone NextGen Server Time Zones....1740
MashZone NextGen Server Administration....1740
View MashZone NextGen Logs....1741
View the MashZone NextGen Server Log....1741
View the Audit Log for a Mashable, Mashup or App....1742
Purge the Audit Log....1743
Manage Pluggable Views and Libraries....1743
Manage Pluggable Libraries....1743
Manage Pluggable Views....1744
Manually Changing the Default Version for Libraries....1745
Manage Files for MashZone NextGen Features or Artifacts....1745
Add External Resources as MashZone NextGen Files....1746
Find MashZone NextGen Files....1746
Update or Delete MashZone NextGen Files....1747
File Organization....1747
Deploying MashZone NextGen Instances, Clusters or Artifacts....1748
Deploying the Core Components....1749
Deploying Server and Client Add-Ons....1750
Deploying MashZone NextGen Artifacts and Other Metadata....1750
Exporting Mashable and Mashup MetaData....1755
Exporting Macros....1756
Exporting App MetaData....1758
Exporting Pluggable Views or Libraries....1759
Exporting MashZone NextGen Global Attributes....1762
Exporting Users, User Metadata and Groups....1763
Importing Mashable or Mashup MetaData....1763
Importing Macros....1764
Importing App Metadata....1765
Importing Pluggable Views or Libraries....1767
Importing MashZone NextGen Global Attributes....1769
Importing Users, User Metadata and Groups....1770
Deploying Multiple MashZone NextGen Servers in One Host....1771
Clustering MashZone NextGen Servers....1771
Setting Up a New Cluster....1772
Adding New Members to an Existing Cluster....1773
Sharing the MashZone NextGen Repository in Clustered Environments....1774
Create and Share a New MashZone NextGen Repository....1774
Share an Existing MashZone NextGen Repository....1776
Setting Up an External MashZone NextGen Configuration Folder....1777
MashZone NextGen File-Based Configuration and Extensions....1778
Sharing JDBC Drivers in Clustered Environments....1783

Additional Information and Support....1787

Preface

The MashZone NextGen User & Developer Guide includes an introduction MashZone NextGen and instructions to MashZone NextGen Hub business and developer users on working with MashZone NextGen to create and publish apps. This guide discusses topics for MashZone NextGen version 9.10.

Additional Information and Support

Samples, Help and Other Documentation

You can find samples, help and other documentation for MashZone NextGen in:

- MashZone NextGen User and Developer Guide: You can download MashZone NextGen documentation at https://empower.softwareag.com/ for offline access. An Empower account is required.
- *Online Help*: is accessible in MashZone NextGen Hub and the AppDepot from *?* help buttons. Help is also available in the MashZone NextGen Add-On for Excel in the toolbar for the add-on.
- *Documentation for Other Software AG Products*: is available online at http://documentation.softwareag.com with an Empower or from http://techcommunity.softwareag.com/welcome-documentation with a Software AG Tech Communities account.
- *MashZone NextGenMashup and Macro Samples*: samples of mashups scripts and macro libraries using the EMML language are available in the Mashup Editor in MashZone NextGen Hub. Samples for RAQL can be found in the *MashZoneNG-install* /prestocli/raql-samples folder.

Version and License Information

You can find version and other system information for MashZone NextGen in the Admin Console.

For system information:

1. Log in to MashZone NextGen and click **Admin Console**.
2. Expand the Server menu section and click **Server Info**.
3. Select the **Platform Information** or **System Information** tabs.

For license information:

1. Log in to MashZone NextGen Add-On for Excel and click **Admin Console**.
2. Expand the Server menu section and click **License**.

Add-ons, Updates, Forums and Technical Support

Find, Contact or Download	From
The MashZone NextGen Add-On for Excel installation package	Software AG Tech Community for MashZone NextGen Add-On for Excel at http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto/downloads.
The MashZone NextGen Add-On for Portals	
Questions and discussion forums	http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto
The MashZone NextGen Add-On for SharePoint	Empower, the Software AG support portal, at http://empower.softwareag.com/.
Technical support, use the eService features.	To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com. For crisis incidents, please call 1-800-525-7859 in the United States. Outside the US, you can find your local access number in the Software AG Global Directory.

1 Introducing Software AG MashZone NextGen

- Real-Time Intelligence 46
- How Does MashZone NextGen Meet This Challenge? 47
- MashZone NextGen Artifacts: A 'Black Box' View of the Flow of Information 49
- MashZone NextGen Core Components 51
- MashZone NextGen APIs, Specifications and Extension Points 58
- MashZone NextGen Add-Ons 63
- MashZone NextGen concepts 66

Real-Time Intelligence

So what is *real-time intelligence*? One useful definition is:

"Insights that provide timely, relevant information with meaningful metrics for real-time decisions"

To live up to this goal, real-time intelligence requires:

Real-Time and Comprehensive Data including:

- Access to live data sources with no lags from intermediate copies or storage requirements.
- Quick and easy access to a wide scope of information sources to ensure information is both relevant and well focused.

  It must be simple to work with information from virtually any source, internal or external, historical, transactional or real-time, regardless of the technical requirements to access the information. And it must support secure access to information sources.
- Flexible support for new information sources to keep pace with the changing 'information landscape'.

  New information sources are created or discovered at an ever-increasing pace, just as new information requirements are.

Meaningful, Actionable Insights with:

- A flexible, but powerful mechanism to combine, transform or analyze information to easily produce metrics and insights.
- Direct leverage of business domain expertise to create metrics and insights and deliver them in rich, visual forms.

  Using domain expertise directly enhances both relevance and timeliness. Information designed 'at the source of the problem' also tends to avoid translation errors common when design or delivery involves cross-domain expertise.
- Easily defined Key Performance Indicators (KPIs) that focus on your operational needs.

  Meaningful KPIs help drive quick, incisive actions to real-time operations.
- Both common and business-specific visualizations to make insights easily understood and compelling.

  Visual forms must include common representations such as tables, maps and charts. This should also be flexible allowing business-specific or other more unique visualizations to be used as needed.

Portable Results:

To provide immediate and simple access to the resulting information, anywhere users need it.

Information needs to be delivered 'everywhere' users go and must be easily, but appropriately shared. Increasingly this means access on mobile devices as well as more traditional desktop environments.

It also means that information must be accessible in a variety of destinations, such as web sites, portals, collaboration systems such as Microsoft SharePoint, wikis or in an internal app store.

End-to-End Security with:

- Secure access that follows IT policies at every step.

  Governance must be applied to both new and old forms of information. User access to enterprise information simply cannot be a free-for-all. As users or developers create new information sources, access controls must also evolve seamlessly.

- No data saved to devices.

  Real-time access ensures no data is persisted on devices potentially allowing unsecured access.

- Flexible security with support for both common and unique security protocols as needed. Information sources and delivery destinations alike may have different security requirements.

How Does MashZone NextGen Meet This Challenge?

To provide the features needed for real-time intelligence, MashZone NextGen contains:

- Tools to quickly make a wide variety of information sources *mashable*. Mashables allow MashZone NextGen to securely connect to information sources and retrieve real-time data on demand.

  Information sources can include systems, applications, databases or common office documents.

- Standards-based integration plus extension points for flexibility to support a wide variety of information sources and security environments.

- Support for streaming and in-memory management to work with large datasets found in non-traditional *big data*, such as social media, monitoring, machine sensor readings or risk management transactions.

- Mashups to provide analysis and data transformation capabilities that handles requirements from very simple to very complex.

  They also provide simple extensions points for business-specific needs.

- A *domain-specific language (DSL)* for mashups to support both simple mashup definition and the breadth of features mashups need.

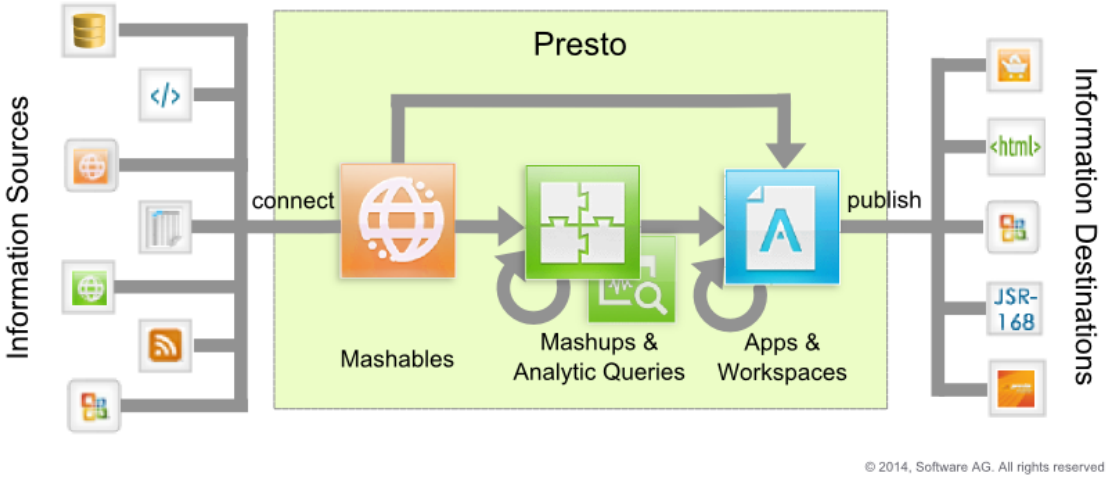
- A simple, but powerful query language to easily work in mashups with large datasets and apply analytics for common business computations, statistical analysis, machine learning or unique analysis needs.

  This includes an extensible list of analytics to cover both common needs and user-defined analytics for unique requirements.
- A wide variety of built-in *views*, the visual form to apply to information, that support both desktop and mobile devices.
- An API and framework to support custom, *pluggable* views that provide different visualizations and leverage third-party visualization libraries.
- Apps as the delivery package to wrap views and mashable or mashup results for easy access to users. Apps can easily be created for desktop browsers and mobile phone or mobile tablets.

  Apps can contain one or several views of the results for one mashable or mashup. *Workspace apps*, or dashboards, combine several apps or views from different mashable or mashup results to work together.
- An API and Specification for apps to support custom requirements that builds on current industry trends.
- An Enterprise AppDepot to use within your enterprise or organization that allows users to easily find and work with apps in desktop browsers.
- MashZone NextGen Mobile apps for mobile phones or tablets to allow mobile users to easily find and work with MashZone NextGen apps in mobile devices.
- Add-on components to support specific information sources, such as spreadsheets, or other app destinations and leverage other technologies such as Microsoft SharePoint or portals.
- Visual tools to quickly and easily create mashups and apps by *power users* (domain experts) and developers.
- Developer tools for mashups, analytic queries and apps to handle more complex problems or build custom plug-ins for visual tools that encapsulate business logic or enable power users.
- A collaborative Hub environment that integrates power user and developer tools to create and share mashables, mashups and apps.
- A server, repository and run-time environment to support creation, management and delivery of real-time operational intelligence.

MashZone NextGen Artifacts: A 'Black Box' View of the Flow of Information

MashZone NextGen directly hosts apps and mashups and acts as a virtual host for mashable information sources. These are the *artifacts* that users work with in MashZone NextGen.



Information Sources

Information sources you can access through MashZone NextGen includes databases, applications or common documents, such as spreadsheets. Information sources may be internal to your organization or external.

Information sources can be accessed by many different protocols, such as the SOAP, REST, RSS or Atom protocols. They can also be file-based in Excel, CSV or XML formats. Or SharePoint lists and searches. They can be databases or datasets stored in memory.

They return information, their results, in many different forms.

Note: Access to some mashable information sources requires MashZone NextGen Add-Ons. Some MashZone NextGen Add-Ons also provide users with access to results from MashZone NextGen mashables or mashups.

Mashables

Users create *mashables* in MashZone NextGen to quickly and securely connect to different information sources and retrieve data as needed. Mashable artifacts define the connection and security configuration and the specific operations or queries needed to retrieve data from databases, applications or documents.

MashZone NextGen simplifies and normalizes results from different information sources for easy access. Users access and run mashables using a single, simple interface without having to manage or understand all the complexities of communication, security and different protocols. MashZone NextGen returns mashable results in either XML or JSON formats.

Mashups

In MashZone NextGen, mashups simplify, combine or transform data to produce just the information you need. Users create *mashups* that combine or transform results from mashables, databases or from other mashups in useful ways.

Mashups can also simply wrap applications to make them accessible in MashZone NextGen. Both mashables and mashups use seamless access policies for governance over all information sources.

Mashups can also work with very large amounts of information, sometimes called 'big data.' Large datasets require a MashZone NextGen Add-On that uses streaming and *analytic queries*.

Analytic Queries

Analytic queries work within mashups to retrieve, analyze and transform large, streaming datasets. Datasets may come from files, databases, information sources with URLs, snapshots of MashZone NextGen mashables or mashups or from datasets already stored in memory.

Analytic queries handle both the streaming nature of large datasets as well as providing *analytic functions* to analyze information using statistics, maching learning or other analysis methodologies.

Views and Apps

Users add *views*, different visual formats for the data, to mashable or mashup results and package the results and views as *apps* that other users can work with. Apps can also be dashboards that combine information from several sources to work together. Or apps can wrap apps, widgets or web pages from other sources.

Apps can be designed for desktop browsers, mobile phones, mobile tablets or any combination of these devices, allowing information to be easily delivered wherever users need it.

The scope of apps includes:

- *Basic apps* which combine one or more views with the results from one mashable or mashup. You can create many different basic apps using different views based on the same results. Or include several different views of the same results in one basic app.

  MashZone NextGen includes a wide variety of built-in views for common formats such as grids, charts and maps. Most built-in views are compatible with both desktop and mobile devices.

Developers can also use the MashZone NextGen View API and framework to easily add other visualizations. These custom, pluggable views can easily leverage the wide variety of third-party visualization libraries.

Note: Licensing requirements by third-parties must also be met.

- *Workspace apps* combine several basic apps, custom apps or views from different mashables or mashups into a composition to work together as a dashboard. Workspaces are also apps and thus can be published to and used in the same destinations.
- *Custom apps* allow developers to meet more complex user interface needs or define more sophisticated behavior when built-in or pluggable views are not sufficient to meet your needs. Developers create custom apps with the MashZone NextGen App Specification and MashZone NextGen App API.

You also apply access policies to apps, just like mashables or mashups, to ensure appropriate governance. Once apps are complete, they are published to destinations as needed.

Information Destinations

Apps can be published to many different destinations where other users can find and work with them. Destinations include two MashZone NextGen components, the AppDepot and the MashZone NextGen Mobile apps, plus external destinations including web pages, Java portals or Microsoft SharePoint.

Note: Some app destinations require MashZone NextGen Add-Ons.

Users can work with desktop-compatible apps in the AppDepot or use the MashZone NextGen Mobile apps in mobile phones or tablets to use apps that are mobile-compatible.

Regardless of the final destination, MashZone NextGen enforces the access permissions you have defined for the app and any mashables, mashups or other apps that it uses. For external destinations, you can also chose to loosen access rules to allow guest users, who are not authenticated, to work with apps.

Next: "MashZone NextGen Core Components" on page 51

MashZone NextGen Core Components

The high-level, core components in the MashZone NextGen platform are two web applications: the MashZone NextGen Hub, where users create dashboards, mashables, mashups and apps; and the AppDepot, where users find and work with published apps in desktop browsers.

In addition, MashZone NextGen contains the MashZone NextGen Server, the MashZone NextGen Repository and the MashZone NextGen Analytics In-Memory Stores to process mashables, mashups and apps at runtime and support creation and management of these artifacts.

Extensions and add-ons for MashZone NextGen are presented later in this introduction.

Tools and Users for MashZone NextGen Hub and the AppDepot

Power users, developers, administrators and finally end users work in the MashZone NextGen Hub and the AppDepot to create, manage, publish and finally find and use dashboards, mashables, mashups and apps.

See also "MashZone NextGen Add-Ons" on page 63, where end users may find and work with mashups and apps in other destinations including mobile apps.

MashZone NextGen Hub

The MashZone NextGen Hub is the collaborative work place for power users, developers and administrators. MashZone NextGen Hub provides a single place to use MashZone NextGen tools to create mashables and mashups, create and publish apps, and manage all of these artifacts. It also provides an administration console for MashZone NextGen administrators to configure and manage the MashZone NextGen Server and MashZone NextGen Repository.

MashZone NextGen Hub users can collaborate with each other by sharing, tagging, rating and commenting on any artifact in the MashZone NextGen Hub. MashZone NextGen Hub also provides a powerful search and browse feature to find any artifact based on common filters or specific search terms and criteria. Community reuse and rating of mashables, mashups and apps maximizes the value of IT assets and improves the quality of information for everyone.

App Maker

The App Maker is a visual wizard for power users or developers to easily create basic apps from any single mashable information source or mashup in MashZone NextGen. No coding required!

Users can choose one or more views from a wide variety of formats for mashable/mashup results. Built-in views include several types of tables, a web feed reader, maps and many types of charts. Developers can also add pluggable views for power users to choose from. Depending on the views chosen, you can design apps for desktop browsers, mobile phones, mobile tablets or any combination.

Note: Most built-in MashZone NextGen views are compatible with both desktop and mobile devices. Pluggable views can also be compatible with multiple devices. Some views, however, are only compatible with desktop browsers and are not shown in mobile devices.

App Maker allows you to preview and update app configuration till you are satisfied with the app for the devices you choose to support for that app. When users open apps, MashZone NextGen automatically displays the views that are compatible with their current device.

Mashboard

Mashboard is a visual, drag-and-drop designer for power users or developers to create *workspace* apps that combine and integrate several basic or custom apps or views to work together in a dashboard or simply group apps and views for a specific focus. Workspace apps can be published and used just like basic or custom apps. This easy-to-use, visual dashboard designer enables you to quickly create powerful apps for even the most challenging business problems and ad-hoc situations.

Note: Some workspace apps are compatible with mobile devices, but some may not be, depending on the layout you choose and the device compatibility or the apps and views you add to the workspace.

Mashboard supports a wide variety of layouts for combining apps including simple columns, tables, drill downs, a desktop layout, multiple tabs or multiple pages. You can also include widgets from other environments or web pages along with apps and views into workspaces for more flexibility.

App Editor

App Editor is a web-based tool for app developers to upload or update custom apps. Custom apps allow developers to create more complex user interfaces or match specific look and feel standards that may not be possible with basic or workspace apps.

Custom apps use a MashZone NextGen App Specification plus common web application resources such as HTML, JavaScript, CSS, images and other media. With the App Editor, you can upload or download all related, client-side resources for an app as a package, making it easy for developers to share app packages or add or update custom apps in MashZone NextGen from their development code base.

Wires

Wires is an easy-to-use, visual drag-and-drop, web-based tool ideal for quick and easy visual mashup creation, no coding involved! Power users and developers can use Wires to create and modify mashups, performing custom, real-time data integration tasks. You can quickly review results at any step within the mashup for immediate feedback.

Wires provides easy access to all available mashables and mashups in MashZone NextGen to use as information sources in a mashup. It includes many built-in, power-packed *action blocks* for easier mashing that leverage the underlying features of the MashZone NextGen platform. Developers can also create custom blocks to extend Wires functionality and handle utility or business-specific requirements. You can also create input parameters for mashups to set properties dynamically when the mashup is used.

Wires simplifies the process of creating mashups, but also limits the mashup features you can use.

Mashup Editor

The Mashup Editor is a web-based authoring tool for mashups and macros defined in the Enterprise Mashup Markup Language (EMML). Developers can rapidly and easily create mashups and macros using the full features of EMML.

Mashup Editor provides easy access to all available mashups and mashables to use as information sources in a mashup. You can also quickly add EMML code, add RAQL queries to work with large datasets or call EMML macros. Mashup Editor allows you to easily run a mashup to test it and inspect the performance characteristics of your EMML statements.

Developers can also create and test EMML macros for use in mashups or for use in Wires as custom action blocks.

Dashboard

MashZone NextGenDashboard gives you an easy, graphic way to create, manage and view your dashboards.

Dashboards are interactive applications that collects data from different data sources, combines it, and visualizes it. The data can come from different sources and is combined. Possible data sources include *Event data source* or *BigMemory data source*. Dashboards are composed of individual components (Real-time Chart, MashZone NextGen App, MashZone NextGen Mashup and MashZone NextGen Grid), which obtain their data from data sources and display it. You can combine the individual components to filter the displayed results interactively and thus analyze them intuitively.

The Admin Console

MashZone NextGen Hub also includes the Admin Console, a web based tool for MashZone NextGen administrators to configure and manage the MashZone NextGen Server. This simple, easy to use tool offers a wide degree of control to inspect, configure, and manage the features of MashZone NextGen.

MashZone NextGen Enterprise AppDepot

Users create apps for both desktop and mobile environments in MashZone NextGen. You can make these apps available to end users by publishing them to the AppDepot or to other destinations. Once published, end users can find and use desktop apps in the AppDepot. For mobile access or other destinations, you must used MashZone NextGen Add-Ons.

The AppDepot makes it easy for end users to find desktop apps that suit their needs and instantly begin using them. If they like an app and anticipate using it frequently,

they can add it to their Favorites gallery for easy access. Favorites in the AppDepot also allows users to customize apps with their own settings as a *personal copy*.

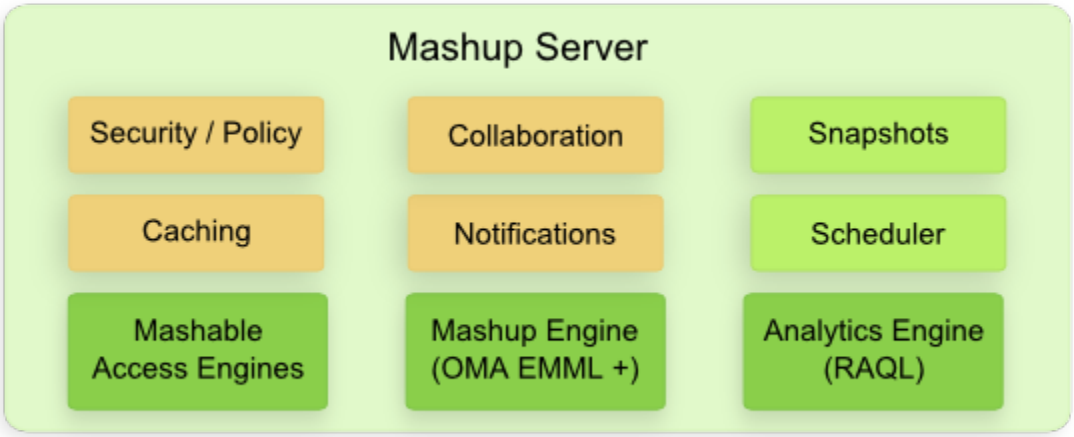
Apps are created by power users and developers in the MashZone NextGen Hub and selectively submitted to the AppDepot. Submitting an app notifies the AppDepot manager (typically an administrator) for review and approval. The AppDepot manager has the ultimate authority to approve or reject an app for the AppDepot. Publishing also makes mobile apps accessible in mobile devices using MashZone NextGen Mobile apps (part of the MashZone NextGen Mobile).

If the AppDepot manager rejects an app, they can send the app back for further development, embellishment and modification. Once any issues have been successfully addressed, app creators can resubmit apps for consideration to be published to the AppDepot.

End users can work with any number of published desktop apps simultaneously in the AppDepot. AppDepot users can share their apps, rate, comment and send feedback to the app creator based on their experience with the app.

MashZone NextGen Server

The MashZone NextGen Server supports design activities, discovery and governance for all MashZone NextGen mashables, mashups and apps.



© Copyright 2014, Software AG. All rights reserved.

- **Mashables**: creating and using mashables uses *mashable access engines* to handle secure access to information sources and results from all types of mashables. Common authentication protocols are supported for mashables, but these can be extended to handle custom requirements.
- **Mashups**: creating and running mashups and macros based on EMML uses the Mashup Engine. MashZone NextGen extends OMA EMML to provide:
  - Secure invocation of mashable information sources, including database mashables.

- Direct invocation of accessible, unregistered information sources.
- Direct invocation of SQL commands in databases.
- Access to large, streaming datasets supported by the Real-Time Analytics Query Language. (Data access support and other features of RAQL are discussed later in this introduction.)
- Execution of RAQL queries to analyze, filter, sort and group large datasets.
- Load or store large datasets and query results in the MashZone NextGen Analytics In-Memory Stores.

The Mashup Engine interprets EMML and includes a scripting engine to handle scripting hooks to custom logic and support for EMML macros. Mashup statements to invoke mashables are handled by the appropriate mashable access engine leveraging the mashable security framework.

The Mashup Engine works in conjunction with the Analytics Engine to handle EMML extension statements that work with large, streaming datasets and run RAQL queries.

- **Apps**: creating and running apps is based on the App Specification. Apps that use mashables and mashups use the Mashable Access Engines or the Mashup Engine for those resources.

The MashZone NextGen Server also provides the following features:

- **Snapshots**: allows users to run mashables or mashups and save snapshots of results for use in analysis or mashups. Snapshots can be user initiated or scheduled to support automatic snapshots for trend analysis.
- **Collaboration and Notifications**: allows sharing between users for discovered artifacts. Both sharing and the app publishing workflow generate notifications.

  Notifications can be sent via email, if mail is configured in the MashZone NextGen Server. Users can also see notifications in the MashZone NextGen Hub home page or My Apps in the AppDepot.
- **Caching**: caching can be configured for mashable and mashup results to enhance performance. Caching configuration uses a flexible inheritance policy to easily define caching requirements for different types of mashables and mashups but also support individual requirements.

  MashZone NextGen has built-in support for clustered caching using BigMemory or memcached.
- **Security**: this includes both authentication and authorization for users when mashables, mashups or apps are viewed or run. The MashZone NextGen Server also handles authentication with mashable information sources when they are run.

  The MashZone NextGen Server is integrated with your user repository or identity server for user authentication. This can be basic authentication, secure connections and certificates or a single sign-on solution.

You define authorization policies for MashZone NextGen resources determining who can view or run mashables, mashups and apps. Generally, users must be authenticated, but you can also define unlimited access, allowing 'guest' users without authentication to work with apps that are published to web sites, wikis or other environments.

Authorization policies also include *entitlements* in MashZone NextGen Hub and the AppDepot. You can determine access to specific tools and features based on authorization policies.

MashZone NextGen Repository

The MashZone NextGen Repository contains meta-data for mashables plus all MashZone NextGen mashups, views and apps, including related files for apps or views, data snapshots for file-based mashables and scheduled or user-initiated snapshots for mashables or mashups. It also contains information on users and groups, authorization policies, server configuration, macros, taxonomies, notifications and much more.

Data can be separated into three parts:

- **Artifacts and Meta-Data**: for mashups, mashables, views, apps, and macros. This also includes related files or data, configuration for the MashZone NextGen Server, authorization policies and so on.
- **User Data**: for authentication and determining authorization.

  Typically user data comes from your organization's LDAP directory which you integrate with MashZone NextGen. This may also use a single sign-on solution and an identity manager. However, MashZone NextGen also has a built-in user repository which you may use. User or group meta-data from LDAP allows MashZone NextGen to relate authorization policies with users.
- **Snapshots**: represent the results from one invocation of a mashable information source or a mashup. Because space and other quality of service requirements can be quite different for snapshot data, this portion of the MashZone NextGen Repository can be hosted in a separate database.

Note: User or mashable passwords stored in the MashZone NextGen Repository are encrypted.

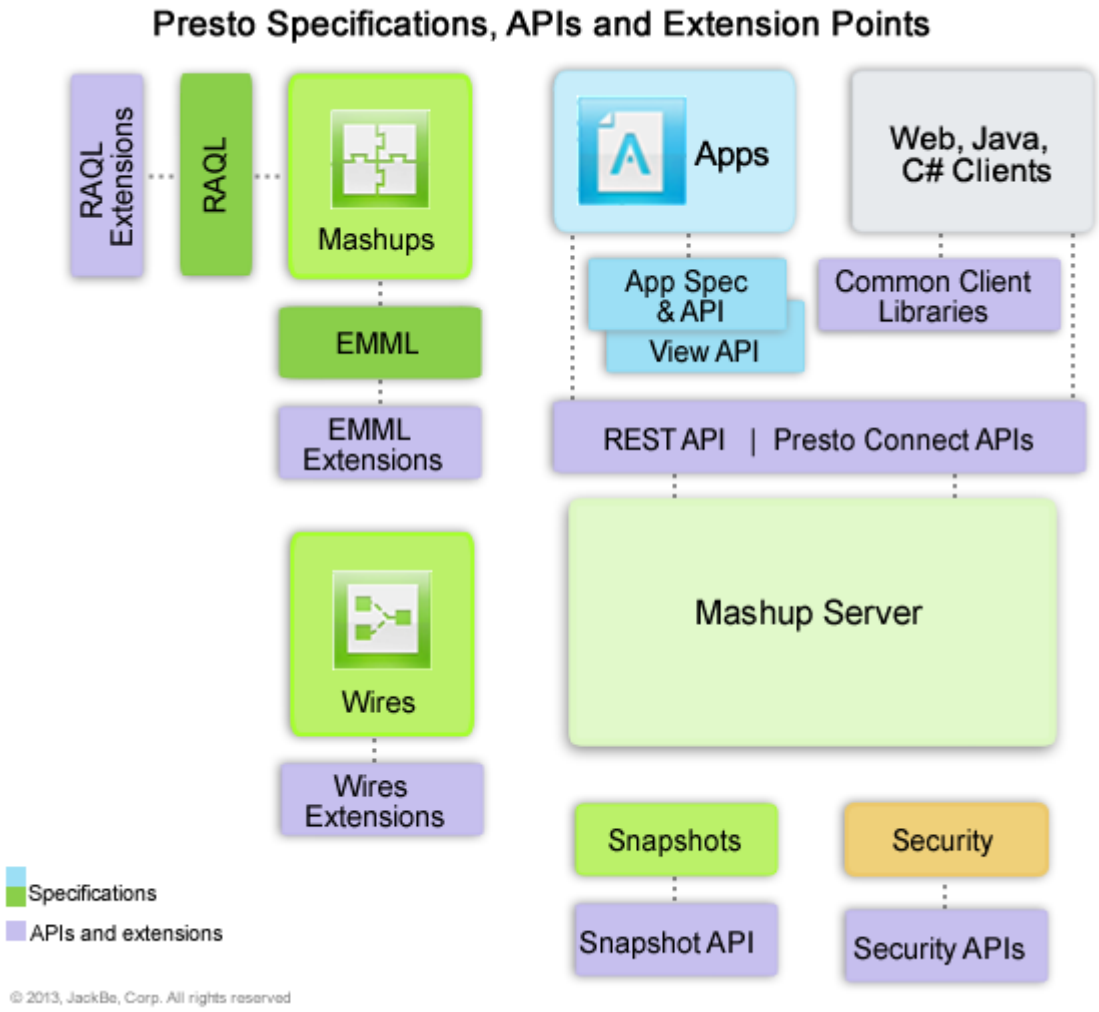
MashZone NextGen Analytics In-Memory Stores

The MashZone NextGen Analytics In-Memory Stores is an in-memory store to support easy, quick access to large datasets for analysis using RAQL. This in-memory store in managed by the BigMemory Add-On. Datasets or query results are stored and retrieved using EMML extension statements in mashups.

Storing datasets or query results provides better performance, in many cases, when the dataset is used in several mashups. It is also useful for datasets with frequent or continuous updates where appending updates is appropriate.

MashZone NextGen APIs, Specifications and Extension Points

MashZone NextGen uses three specifications to describe mashups and apps: the Enterprise Mashup Markup Language (EMML), the Real-Time Analytics Query Language (RAQL) and the App Specification respectively. In addition to these specifications, MashZone NextGen includes several APIs and extension points shown in the following diagram:



These APIs and extensions provide the flexibility you need to:

- Create mashups, views and apps that fit your unique information requirements.
- Add domain-specific enhancements to simplify mashup creation and empower power users.
- Add user-defined analytics to meet domain-specific or unique analysis needs.
- Use a wider variety of mashable information sources or work with information sources with unique security requirements.
- Customize authentication for MashZone NextGen.

Enterprise Mashup Markup Language: The Mashup Domain Language

The Enterprise Mashup Markup Language (EMML) is the domain language for mashups from the Open Mashup Alliance (OMA). MashZone NextGen provides a full implementation of EMML and extends its capabilities to work with all types of MashZone NextGen mashables and mashups within MashZone NextGen governance as well as large, streaming datasets using RAQL.

The EMML DSL provides a simple, declarative vocabulary in XML. You create *mashup scripts* using MashZone NextGen tools that provide a visual or code perspective of the mashups script. Mashup scripts are written in EMML that is interpreted by the MashZone NextGen Server at runtime.

EMML in MashZone NextGen provides simple, straight-forward semantics to address a wide range of mashup requirements, including:

- Invoking MashZone NextGen mashable information sources or other mashups, within defined access policies.
- Invoking any web-based service or URL where access should not be limited.
- Invoking SQL commands directly for multiple database tables in a single datasource.
- Loading, storing or querying and analyzing streaming datasets using the Real-Time Analytics Query Language .
- Defining variables and parameters with simple types or complex types. Complex types are represented as well-formed XML *documents*, allowing any level of structure or hierarchy needed.
- Variable scoping for flexibility.
- Constructing variables and results with static or dynamic content. Selecting specific records or specific columns.
- Combining data that is structurally identical (a merge) or that shares a relationship (a join).
- Providing common flow-control logic such as for loops, if-then-else and parallel or sequential flows.

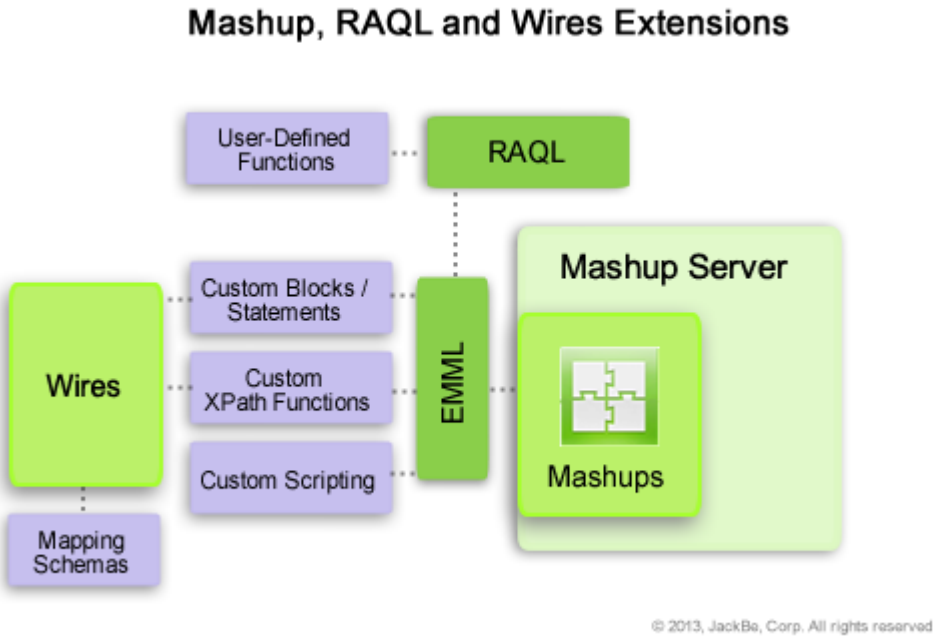
- Transforming data including:
  - Sorting.
  - Filtering.
  - Grouping.
  - Assigning simple values or whole or partial complex structures.
  - Annotating to add data or meta-data to existing structures.
  - XPath functions, both standard and user-defined for data conversions, validations, calculations and formatting.
  - XSLT stylesheets to handle complex transformations or leverage existing logic.
- Providing character encoding support and CDATA escaping.
- Handling transactions, both for database mashables and direct SQL commands.

  Note: EMML does not support distributed transactions.
- Including metadata, both user-defined and defined by MashZone NextGen.
- Supporting debugging needs with assertions or dynamic messages.
- Supporting exception handling with try/catch.
- Allowing dynamic expressions and templating for EMML properties, including XPath expressions.
- Extension points to easily support domain-specific and custom logic at different levels within mashups. See "Mashup Extensions in EMML" on page 60 for details.

Using an XML vocabulary allows EMML to be easily written by tools or off-the-shelf editors. It builds on the XPath, XQuery and XSLT standards for strong query and transformation features.

Mashup Extensions in EMML

EMML supports several extension points for mashups.



This includes:

- Mashups can themselves be reusable logic designed to provide common functionality to other mashups.

  For example, a mashup can log in to a specific mashable and provide authentication tokens or cookies as output to any mashup that needs to work with that mashable.

- Macros define custom statements in EMML. They accept input, provide output and can include any mashup logic, including defining and calling other macros.

  Macros can be used in just one mashup or macro. Or they can be published in libraries for use in any mashup or macro. Macros also provide an extension point for Wires.

  Note: MashZone NextGen registers macros in the MashZone NextGen Repository just as it does with mashups to support easier deployment and straight forward visibility in clustered environments.

  For example, a macro can perform a domain-specific weighted average calculation or can encrypt/decrypt data for a specific encryption algorithm.

- Hooks to call scripts in Groovy or JavaScript or to access Java classes. This provides direct, easy access to existing logic or enables custom logic using familiar languages.

  Scripting does have an impact on performance. In many cases, a custom XPath function is a better solution.

- Custom XPath functions written with the MashZone NextGen Custom XPath API are particularly useful for logic that transforms data or perform calculations or validation. XPath functions also provide better performance than scripting.

You can use these extensions in Mashup Editor, the EMML code editor. Many of these extensions can also be used in Wires, the visual tool editor.

Real-Time Analytics Query Language

The Real-Time Analytics Query Language (RAQL) is an easy-to-learn, SQL-like language to query and analyze large, streaming datasets. Users access datasets in mashups using EMML extension statements that leverage RAQL.

RAQL gives you direct, streaming access to data:

- In files, without registering them as MashZone NextGen mashables.
- In databases or from URL-addressable services.
- In any number of snapshots for MashZone NextGen mashables or mashups.
- Stored in the MashZone NextGen Analytics In-Memory Stores.

  Extension statements in EMML also let you store large datasets or query results in the MashZone NextGen Analytics In-Memory Stores.

You can also use RAQL with results for any MashZone NextGen mashable or mashup.

RAQL works with data in CSV, XML or JDBC result formats. Data should be in a flat hierarchy, with simple columns and rows. RAQL queries can:

- Filter dataset columns and rows using simple or complex criteria.
- Sort data with any number of levels.
- Group data in groups, partitions or windows and analytic functions to these sets.
- Use subqueries and many other common SQL features to handle complex requirements.
- Use the plain and analytic functions that are built into MashZone NextGen plus any extensions that user define.

RAQL Extensions: User-Defined Functions

In addition to the RAQL functions built into MashZone NextGen, you can define additional functions to use in RAQL queries to handle unique or domain-specific needs. User-defined functions can be plain functions. Or they can be aggregate or window analytic functions that use the MashZone NextGenRAQL User-Defined Function API.

You add user-defined functions in libraries with library names that you assign. This allows you to manage functions and ensure that function names are unique.

View API and Framework

Basic apps use views that have been configured for a specific mashable or mashup using the View Maker wizard in MashZone NextGen Hub. MashZone NextGen has a set of built-in views that appear in View Maker which users can choose from. You can also use the MashZone NextGen View API and framework to plug *pluggable views* with other visualizations into View Maker to meet your visualization needs.

This API allows developers to quickly implement pluggable views using JavaScript, HTML and CSS resources which are imported into MashZone NextGen much as custom apps are. Pluggable views render mashable or mashup results and also contain a configuration form that plugs into the well-known steps in View Maker to allow users to easily configure the view. You can provide a thumbnail image and help for pluggable views as well as managing versions and other properties. Pluggable views also use the MashZone NextGen DataTable API, discussed later in this introduction, as the client model for mashable or mashup results.

In addition to adding pluggable views, this framework allows developers to add pluggable libraries with JavaScript, HTML and CSS resources that can be used to support pluggable views or custom apps. Pluggable libraries may be specific to your organization or they can be from third parties. They can be hosted in MashZone NextGen or hosted externally. This feature makes it easy to leverage the wide variety of web visualizations available.

Note: You must meet licensing requirements for any third-party libraries that you use in pluggable views or custom apps.

MashZone NextGen Add-Ons

Generally, MashZone NextGen Add-Ons allow you to work with less traditional information sources, such as spreadsheet documents, or to publish MashZone NextGen mashups and apps to mobile devices or portals such as Microsoft SharePoint or Liferay for end user access. Add-Ons provide additional connectivity and integration.

MashZone NextGen Real-Time Analytics Add-On with Terracotta BigMemory

This MashZone NextGen Add-On, unlike other add-ons, is not a destination for apps nor an alternate source for information for MashZone NextGen. Instead, it enables MashZone NextGen to handle business problems involving 'big data,' such as social media sentiment trends, machine-to-machine interactions, operations monitoring or risk management for real-time transactions.

This add-on:

- Enables MashZone NextGen Analytics, a new component in MashZone NextGen, to load large datasets from different data sources and use analytics within mashups to quickly analyze and derive meaningful insights.

  MashZone NextGen Analytics includes RAQL to query large datasets and apply analytics, as well as extensions to EMML to work with streaming datasets. RAQL also supports access to a variety of data sources including files, snapshots from mashables or mashups and databases.

- Adds Terracotta™BigMemory to provide the straight-forward, extensible, fast in-memory data management needed to work with large datasets.

  BigMemory is available in MashZone NextGen, with limited memory, to let you evaluate MashZone NextGen Analytics and to support the MashZone NextGen Analytics In-Memory Stores and MashZone NextGen caches. The BigMemory Add-On allows you to extend available memory to support your full requirements for large datasets by installing BigMemory as a separate server, hosted externally from MashZone NextGen. You can easily scale available memory with BigMemory servers, handle distributed caching for MashZone NextGen clusters and use robust production features such as failover and a management console.

MashZone NextGen Mobile

The MashZone NextGen Mobile enables mobile features in MashZone NextGen allowing users to easily create views and apps that are *both* desktop- and mobile-compatible.

This add-on also includes the MashZone NextGen Mobile apps that are 'mobile editions' of the AppDepot. (See " MashZone NextGen Enterprise AppDepot" on page 54 for details.) MashZone NextGen Mobile apps are free mobile apps users can download to mobile phones or tablets and easily and securely connect to any number of MashZone NextGen Servers to find and work with mobile apps hosted in MashZone NextGen.

Note: Currently, MashZone NextGen Mobile apps are only available for iOS phone or tablet devices. Users can download MashZone NextGen Mobile from the Apple App Store.

In MashZone NextGen Mobile, users can find apps that are mobile-compatible and that have been published to the AppDepot. Users can only open and work with mobile-compatible apps that they have permission to use. Users can add mobile apps to their favorites in MashZone NextGen Mobile to quickly find apps they use frequently.

MashZone NextGen Add-On for SharePoint

The MashZone NextGen Add-On for SharePoint can act as both a source of information for MashZone NextGen and a destination for apps and mashups. You use this add-on to create native SharePoint Web Parts for apps or mashups, providing dynamic, real-time

integration to the full range of information sources in MashZone NextGen. This also leverages MashZone NextGen governance for these resources.

Once apps or mashups are published as native Web Parts, they can be added to any SharePoint Web Part page for end user consumption. Being native, these Web Parts have deep integration with the SharePoint platform, including support for the SharePoint communication framework. Similar to app integration in workspace apps, this allows Web Parts for Apps and mashups to interact with other artifacts in a SharePoint Web Part page, including other native Web Parts such as SharePoint Lists, Choice Filters, or Text Filters.

MashZone NextGen Add-On for SharePoint also allows data from SharePoint to be used as information sources for MashZone NextGen artifacts. You can register SharePoint lists as mashable information source or directly use SharePoint lists, list views or search results as information in mashups.

Administrators can connect any number of SharePoint farms, sites or site collections with any number of MashZone NextGen Servers and determine the specific scope of each connection. Administrators can also manage authentication requirements in both directions between SharePoint and MashZone NextGen. Generally authentication is session-based, but can also use single sign-on services for the Microsoft SharePoint Server.

MashZone NextGen Add-On for SharePoint supports SharePoint 2010 with Firefox, Chrome, and MS Internet Explorer 10 and higher. If you want to use the SharePoint add on with SharePoint 2010 and Internet Explorer Versions 7, 8 and 9 you have to force SharePoint 2010 to use the Internet Explorer 10 mode. For this you need to update your master page to set the version of Internet Explorer to 10 or above using an appropriate meta tag, e.g. <meta http-equiv="X-UA-Compatible" content="IE=10" />.

MashZone NextGen Add-On for Excel

The MashZone NextGen Add-On for Excel allows worksheets in your organization to become information sources in MashZone NextGen and to become destinations for mashup and mashable results from MashZone NextGen.

This add-on adds a toolbar to Excel that allows you to import results from MashZone NextGen mashups or mashables to a worksheet so that you can work with it. You can refresh results in your worksheet, as needed. Access to MashZone NextGen mashables and mashups is secure, using MashZone NextGen policies.

The MashZone NextGen Add-On for Excel also allows you to register Excel worksheets as MashZone NextGen mashable information sources to allow worksheet data to be used in mashups or apps or to allow other Excel users to share worksheet data by importing the mashable into their own worksheets.

MashZone NextGen mashables may have a simple snapshot of worksheet data or they can have a dynamic connection for worksheets hosted in SharePoint or other document management systems that support HTTP access. For mashables with a snapshot of data, you can update mashable data as needed using this add-on.

MashZone NextGen Add-On for Portals

The MashZone NextGen Add-On for Portals is a *portlet producer* which you integrate with your portal application. This add-on simplifies the integration process between your portal application and MashZone NextGen, allowing you to quickly add and work with MashZone NextGen mashups and apps in your portal.

Integrating MashZone NextGen mashups and apps into a portal environment provides several benefits:

- A simple, but powerful way to consume many common information sources in your portal, including web services, web feeds, relational databases or common business documents
- A simple, but powerful mashup capability to filter, combine and transform information to fit your unique needs
- A simple way to govern access to information provided by MashZone NextGen mashups and apps

The MashZone NextGen Add-On for Portals supports Java Portlet standards. This add-on can be hosted with your portal server or it can be hosted separately, if the portal server supports WSRP.

MashZone NextGen concepts

What is ...? Why can't I ...?	Wires concepts
"What is a mashup?" on page 68 "What kinds of information can I use in a mashup?" on page 68 "What is an artifact?" on page 69 "What are valid artifact names and IDs?" on page 69 "What is a mashable?" on page 68 "What are all the strange input parameters, such as app-id, for mashables or mashups? " on page 69 "What are results?" on page 69 "What is a view?" on page 70 "What is an app?" on page 70	"What is a block?" on page 71 "What kinds of actions can I use in a mashup?" on page 72 "Can I control whether a block is used in a mashup?" on page 72 "When I try to draw connections between blocks they just disappear. How do I connect blocks in Wires?" on page 72 "Why can't I preview results for a block? or my mashup?" on page 72 "I get the wrong results when I use the Sort action with dates. Why doesn't this work?" on page 72

What is ...? Why can't I ...?	Wires concepts
■ "What is a workspace?" on page 70 ■ "What is a snapshot?" on page 71 ■ "What are MashZone NextGen attributes?" on page 71 ■ "Why can't I find, create, run, register or publish ... in MashZone NextGen?" on page 71 ■ "Why can’t I use the artifact name in EMML or other code?" on page 71	■ "Why can't I open and edit some mashups in the Block Menus?" on page 73 ■ "What do all the strange block properties, such as <appid>, mean for a mashable? " on page 73 ■ "What are the "*:" symbols that show up in paths to fields?" on page 73 ■ "What is a namespace?" on page 73

View concepts	Mashboard concepts
■ "None of the views I see display results the way I want. Are there other views I can use?" on page 73 ■ "What is a category?" on page 73 ■ "What is a series?" on page 74 ■ "What are events in views?" on page 74	■ "How do I make one app or view react to user actions in another app or view in my workspace?" on page 74 ■ "What are events in workspaces?" on page 74 ■ "What is wiring?" on page 75 ■ "Why can't I find any published events for apps or views in my workspace?" on page 75 ■ "What events can I use when wiring apps and views in a workspace?" on page 75 ■ "I added an app to a workspace in Mashboard and it has stopped working. I cannot delete the app from my workspace. How do I get rid of this app?" on page 75

What is ...? Why can't I?

What is a mashup?

Mashups take one or more sources of information, combine them, add to them or changes them to quickly produce a *result* that you or other users find useful. For example:

- A mashup receives news articles from Yahoo and removes any articles that do not mention the Dow Jones from its result.
- A mashup combines customer information with weather or other local information based on customer addresses.

In Wires, you define what information you need in the mashup and how it should be changed with a simple set of *blocks*. Each block has a *result* that shows the effect of that block. The result from the entire mashup is shown in the Output block. See also "What is a block?" on page 71 and "What are results?" on page 69.

With EMML, you define the information and processing for the mashup with EMML statements. EMML is an XML languages that provides a rich programming capability to create mashups and extension points to handle a wide variety of requirements. You use the Mashup Editor to write mashups in EMML.

What kinds of information can I use in a mashup?

You can use other mashups or mashable information sources available in MashZone NextGen. If this is enabled, you can also connect to information that is publically available on the Internet or from customer or partner sites.

See "What is a mashable?" on page 68 for more information on all the possible types of information.

What is a mashable?

MashZone NextGen mashables connect to information sources within your organization or sources that are publically available on the Internet and retrieve information. In some cases, mashables connect to an information source and register subscriptions so that information is then published automatically to MashZone NextGen.

Information sources may be:

- Databases, web services, web feeds, applications or documents in your organization or from customers or partners.
- Process, market or other update events generated by business process tracking or other business intelligence processes. Event mashables are also sometimes called *event sources* .
- Publicly available information, such as news feeds, weather information, search and query capabilities for various organizations and many more.

Typically, MashZone NextGen administrators and MashZone NextGen developers register mashables to make these information sources easily available. MashZone NextGen administrators also define permissions for mashables to determine who may see and use them.

What is an artifact?

Artifacts are the mashables, mashups or apps that you or other users create and register in MashZone NextGen.

What are valid artifact names and IDs?

Mashable, mashup and app names in MashZone NextGen can contain characters from the character sets supported by the MashZone NextGen Repository, numbers, spaces, tabs, line ends and these common symbols: _ ~ - * ' . MashZone NextGen uses the name that is first assigned to an artifact to generate a unique ID. IDs contain only characters from the character sets supported by the MashZone NextGen Repository, numbers and underscores (_).

Artifacts can have duplicate names, but artifact IDs are always unique. Owners and MashZone NextGen administrators can change artifact names, but IDs are permanent.

What are all the strange input parameters, such as *app-id*, for mashables or mashups?

Mashables represent an underlying information source which may be designed by other groups with a more technical perspective. In many cases, you don't have to supply values for these input parameters. In some cases, MashZone NextGen developers define default values. Or MashZone NextGen administrators may set up MashZone NextGen attributes to provide this information (see "What are MashZone NextGen attributes?" on page 71).

Try leaving anything confusing blank. If you still have problems, talk with your MashZone NextGen administrator or developers for more information on confusing input parameters.

What are results?

Results are simply the raw information from a mashup, or in Wires from an individual *block*. So results from a mashable or mashup are the information from that source when you run the mashable or mashup.

In Wires, the result for actions and other blocks shows the affect of that action on the information that is linked to that action. The mashup result comes from applying actions to information sources in the mashup in the way that you define.

Wires displays results in a few very simple views. But you can also add many other views to a mashable or mashup. See also "What is a block?" on page 71 and "What is a view?" on page 70.

What is a well-formed document?

Mashups work with the data from information sources as XML. In most cases, this data is *complex*, containing more than a single field. In XML, the structure and relationships in complex data are represented as a *document*. Documents must be *well formed* in order for mashups to work with the data.

Well-formed documents, in XML, follow a set of simple rules. Each document is completely wrapped in a root element or node indicating the beginning and end of the document. The data within the document is represented as elements or *nodes* that are children of the root node. Children may themselves contain grandchildren and further descendants as needed to represent the full structure and contain all the data.

What is a view?

A view defines a particular format to apply to results so that it easier to understand the information or to allow you to see the results in different contexts. For example, views can display results in paragraphs, a table or in a map.

Views can also provide additional behavior. For example, views can create a link that you can click to see more information based on a web address in results.

MashZone NextGen has a set of built-in views that you may use with mashups or mashables to view or share information or create *apps*. MashZone NextGen developers can also define additional, pluggable views for your use.

What is an app?

Apps combine MashZone NextGen mashups or mashables with one or more views into a *plug-in* that you can use in MashZone NextGen *workspaces* or in any web page. Apps are one way for you or other users to work with and share MashZone NextGen mashups and mashables. You can add apps to MashZone NextGen workspaces or to pages in your intranet, portal, SharePoint or wiki.

In previous releases, apps were also called mashlets.

What is a workspace?

Workspaces allow you to easily add and arrange apps into categories and tabs that are useful to you for your daily work. Apps in workspaces can also be synchronized to work together to show detailed or related information.

One common use for workspaces is as a dashboard.

You create workspaces in MashZone NextGenMashboard. MashZone NextGen developers or other MashZone NextGen users can create workspaces for your use.

To use a workspace, it must be published. You use workspaces in the MashZone NextGenAppDepot, MashZone NextGen Mobile, SharePoint, your portal or any web page.

What is a snapshot?

A snapshot is the results from a mashable information source or a mashup when it was run at one specific time. You can take snapshots of any mashable or mashup that you have permission to run. You can also create a schedule to take snapshots automatically.

You can register a snapshot as a mashable to use that snapshot as the basic for a basic app. MashZone NextGen developers can also use snapshots directly in custom apps.

What are MashZone NextGen attributes?

MashZone NextGen attributes are typically values for mashups or mashables parameters that your MashZone NextGen administrator defines. This makes the values available to all users, if they are defined as global attributes or as *flex attributes*. Or different users may have their own unique value, if they are defined as user attributes.

Common examples might be a user name or password that everyone should use to work with a specific mashable. In some cases, MashZone NextGen attributes are technical information that a mashable needs to work properly.

Why can't I find, create, run, register or publish ... in MashZone NextGen?

The most common reasons you may not be able to find mashables, mashups or apps or see menu options, buttons or other features mentioned in MashZone NextGen help or documentation include:

- Your permissions in MashZone NextGen do not allow this. Your permissions define which artifacts you can see and use and which features and actions you can perform.

  In most cases, you simply cannot see the artifact, menu or button. Or you may receive a message indicating that you do not have permission.

  If you feel you should have this permission, contact your MashZone NextGen administrator to request this.

- The license for MashZone NextGen for your organization has specific limits on the features that MashZone NextGen supports.

Why can’t I use the artifact name in EMML or other code?

Duplicate artifact names are allowed in MashZone NextGen. Also both owners and MashZone NextGen administrators can artifact names.

Because of this, references to mashables or mashups in EMML or references to any artifact in code of any kind use the artifact ID.

Wires concepts

What is a block?

Blocks appear in the Wires design canvas as icons that represent either the sources of information or the actions that you want to use to change this information into the result

for your mashup. You connect blocks to define how information is changed to become the final result for the mashup. See also "What is a mashup?" on page 68.

What kinds of actions can I use in a mashup?

MashZone NextGen has a set of built-in actions that allow you to combine, filter or select, add or transform information for the mashup. Actions also allow you to define input fields for the mashup for dynamic results. You change the input when you use the mashup to receive different result.

MashZone NextGen administrators or developers can also define custom actions and add them to Wires. Developers should see "Adding Custom Blocks to MashZone NextGen Wires Using Macros" on page 392 for more information.

Can I control whether a block is used in a mashup?

You can define one or several conditions that must be met before any block is used in the mashup. These are called *execute conditions* that you define as properties of the block.

When I try to draw connections between blocks they just disappear. How do I connect blocks in Wires?

You must draw connections between *ports* of two blocks. Ports are the small hubs or buttons on the sides of blocks. Each block has an input port on the left side of the block, to receive data, and an output port on the right side of the block to send data to another block.

See "Connect Mashup Blocks" on page 383 for complete instructions.

Why can't I preview results for a block? or my mashup?

The most common reason why preview will not work is that one of the mashables or mashups in your mashup is inactive. Inactive mashables and mashups have an inactive flag in the Block Menus and the block shows an inactive flag in the Design Canvas. The service or mashup must be activated before you can preview results for the block or for your mashup.

I get the wrong results when I use the Sort action with dates. Why doesn't this work?

The most common reason for sorting problems for fields that contain dates is that the dates use a format that MashZone NextGen does not support. MashZone NextGen supports U.S. date formats including:

- MM/DD/YY or variations, such as MM-DD-YYYY or MM.DD.YY. For example, 01/05/07 for January 5, 2007.
- Month names (either as three-character abbreviations or completely spelled out) DD, YYYY. For example: Jan 12, 2005 or August 9, 1899.
- Day of the week, Month DD, YYYY [AD|BC]. For example, Tuesday, April 8, 2008 AD.

Why can't I open and edit some mashups in the Block Menus?

Wires only allows you to work with mashups that you create - you cannot edit mashups that other users have created. They are available in the Block Menus so that you can use them in your own mashups.

Wires also will not open advanced mashups, created by MashZone NextGen developers, because these mashups use features that Wires does not support.

What do all the strange block properties, such as <appid>, mean for a mashable?

See "What are all the strange input parameters, such as app-id, for mashables or mashups? " on page 69.

What are the "*:" symbols that show up in paths to fields?

This is a wildcard that allows any namespaces that may be used in results. See "What is a namespace?" on page 73 for more information.

What is a namespace?

A namespace is a unique identifier for a set of XML elements. Namespaces are used in the results for some mashables to allow elements from different organizations to work together without having duplicate names - even if two groups use the same element name, the namespace uniquely identifies each one.

Wires handles namespaces for you so that you do not need to see them or know what they are. They appear in element names as *prefix* :*element-name* . The prefix is actually an alias for a URI (unique resource identifier), such as the URL for a web site.

View concepts

None of the views I see display results the way I want. Are there other views I can use?

MashZone NextGen developers can define additional, pluggable views for your use with the "Template View" on page 1006.

What is a category?

In chart views with both an X and Y axis, such as a line or bar chart, the category is a field that occurs in each repeating item in your results that should be used as the measurement or grouping points (the *ticks*) along the X axis.

For example, you want a chart of revenue for three clients for the last six months:

- Each month of data is one repeating item or *row* in the results.
- The revenue for each client is one field or *column* in each row.
- There is also a field in each row with the name or number of the month. This field becomes the category for the chart.

The category is typically a field with a text value, although this can be a date or a number.

What is a series?

In many chart views, a series represents one field that occurs in each repeating item in your results that should be graphed or charted. For example, in bar or line charts, each series represents the data for one set of bars or one line. Series are also sometimes simply called *columns.*

Series are always numeric fields.

What are real-time charts?

Real-time charts provide common chart views, such as a line chart, *only* for *event mashables*. See "What is a mashable?" on page 68 for more information.

Event mashables subscribe to updates about business process or business intelligence events. Once a subscription is registered, event data is sent to MashZone NextGen automatically as events occur. Unlike other MashZone NextGen built-in views, real-time charts support receiving these ongoing event updates and dynamically updating chart plots.

You also add real-time charts differently than most views. Real-time charts are not available in the View Maker wizard and cannot be added to event mashables from their artifact pages. Instead, you add real-time charts to workspaces in Mashboard and then identify which event mashable should be used as the information source for this real-time chart.

What are events in views?

Events are typically user actions that a view recognizes and publishes to other apps when they are both used in a workspace created in Mashboard. Common examples of events include clicking buttons, selecting rows, selecting bars, pie slices or other graphic regions or entering input parameters.

Each view determines what events it supports. See documentation for the built-in MashZone NextGen views for details.

Mashboard concepts

How do I make one app or view react to user actions in another app or view in my workspace?

This is called *wiring* or sometimes *inter-app communication*. You wire some apps or views to react to *events* in other apps or views. See "What are events in workspaces?" on page 74 and "What is wiring?" on page 75 for more information.

What are events in workspaces?

Events typically happen when users perform some action in an app such as clicking a button, selecting a row or entering a parameter.

Each app or the views that are used in an app define what events they can react to and publish messages for. When an event occurs (a user performs the appropriate action), the app publishes that event which sends a message to all other apps in the same workspace that have subscribed to that event.

What is wiring?

When you wire a workspace, you register subscriptions for some apps or views to the events that other apps or views in the workspace may publish. You may also map the fields from the published message to specific fields that the subscribing app or view is expecting.

Why can't I find any published events for apps or views in my workspace?

Mashboard cannot see events for some of the built-in views used in basic apps, such as charts, until you perform the action that triggers that event such as clicking on a bar.

Try clicking objects in the app or view that should be publishing events. See documentation for the built-in views to help determine what events may be possible.

What events can I use when wiring apps and views in a workspace?

For basic apps, the views that are used in the app define exactly what events that view publishes. In many cases, the events are shown automatically when you wire the workspace in Mashboard.

For some built-in MashZone NextGen views, you may need to perform the user action to ensure that Mashboard recognizes an event. See documentation for the built-in views for event information.

Custom apps and views that a MashZone NextGen developer has created for your organization define what events they publish. If no events are shown in Mashboard, check the description for the app or view, contact the developer who created the app or view, or if you are a MashZone NextGen developer, view the App Specification or view implementation class to determine what events this app or view can publish.

I added an app to a workspace in Mashboard and it has stopped working. I cannot delete the app from my workspace. How do I get rid of this app?

In rare cases, an app conflicts with Mashboard or with other apps or views in your workspace and this causes the workspace to freeze. Since Mashboard remembers the state of the workspaces that you have open, you cannot open this workspace without it freezing.

You must manually reset Mashboard in this case, by adding a `reset` parameter to the URL for Mashboard in your browser. If the normal URL for Mashboard is:

```
http://localhost:8080/presto/hub/mashboard.html
```

Then enter:

```
http://localhost:8080/presto/hub/mashboard.html?reset
```

2 Working in MashZone NextGen Hub

- MashZone NextGen Hub Features 78
- Your MashZone NextGen Hub Profile 82
- Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub 84
- Common Tasks for Mashables, Mashups and Apps 85
- Update Mashables, Mashups or Apps in Bulk 97
- Descriptions, Providers, Categories and Tags for Artifacts 101
- Grant Permission to Run Mashables, Mashups and Apps 102

You work in MashZone NextGen Hub to register, create, find, share and work with mashable information sources, mashups and apps. These *artifacts* let you quickly gain insight into problems, work easily with disparate applications and information, or solve problems that are unique to you or your team. You publish apps to make them available to other users in the AppDepot, your portal, web sites, SharePoint or other environments.

The basic pages and features that you use while you work in MashZone NextGen Hub are shown below.

MashZone NextGen Hub Features

You start working in MashZone NextGen Hub from the Home page using the main menu to find and open dashboards, mashables, mashups or apps or to open tools to create or register various artifacts.

Important: Some MashZone NextGen Hub features discussed in this topic may not be visible to you, based on your MashZone NextGen permissions.

The MashZone NextGen Main Menu includes:

- *Home*: to return to "The Home Page" on page 81 for MashZone NextGen Hub.
- AppDepot: to find and work with published apps for desktop browsers. See "Working in the MashZone NextGen Enterprise AppDepot" on page 1186 for videos and links to more information.

  Note: Users can also find and work with published mobile apps using **MashZone NextGen Mobile**, a 'mobile edition' of the AppDepot for mobile phones or mobile tablets. See "Working in MashZone NextGen Mobile Apps" on page 1189 for videos and more information.

- *Search*: to find and open the artifacts you want to work with. See "Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub" on page 84 for more information.

  You can also use this page to select multiple artifacts and apply updates in bulk. See "Update Mashables, Mashups or Apps in Bulk" on page 97.

- Tools to create, register or edit dashboards, mashables, mashups or apps:
  - **Visualize >** Mashboard: to create *workspaces* that group apps and can also wire apps to work together. Workspaces are also a type of app. See "Create Workspace Apps with Mashboard" on page 1116 for more information.
  - **Visualize >** App Editor: to develop custom apps. See "Create Custom Apps from the Base App Package" on page 1226 for more information.

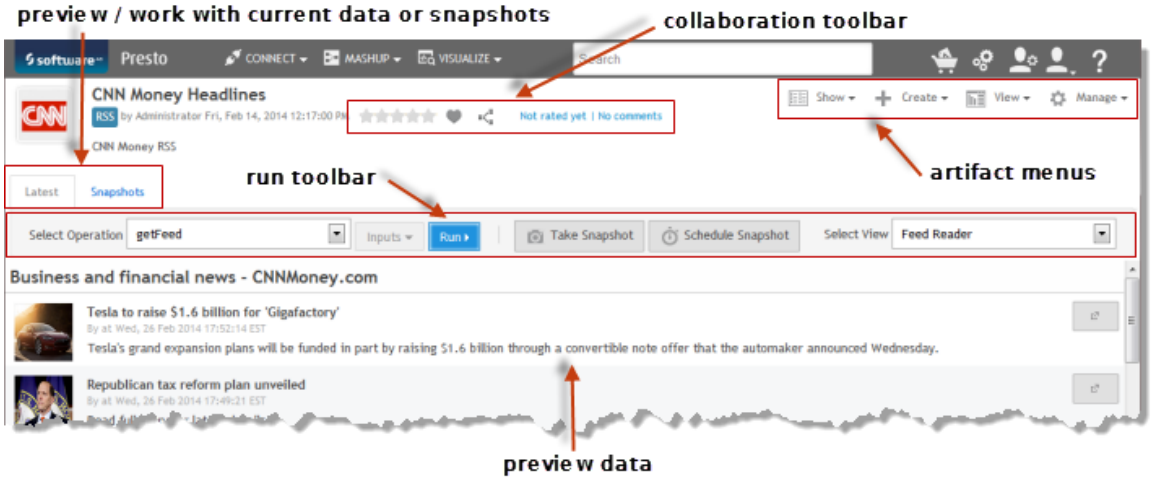
- **Mashups >** Wires: to create mashups visually, using simple drag-and-drop blocks and wiring them together. See "Mashups in MashZone NextGen Wires" on page 281 for more information.
- **Mashups >** Mashup Editor: to create or edit mashups or macros using the Enterprise Mashup Markup Language (EMML). See "Mashups in EMML" on page 441.
- **Connect**: to register mashables that can connect to information sources. See "Connect Information Sources as Mashables" on page 136 for links to more information.
- **Dashboard**: to create dashboards visually. Dashboards are interactive applications that collects data from different data sources, combines it, and visualizes it. See "Dashboards" on page 187.

Each artifact that is created or registered has an *artifact page* where you work with that artifact. You open "Artifacts and Artifact Pages" on page 79 from search results, from your favorites list or other links and work with them or create basic apps from them.

- Admin Console: to manage and configure MashZone NextGen.
- **API Console**: a utility for MashZone NextGen developers or administrators to view and test MashZone NextGen platform APIs. MashZone NextGen administrators and developers should see "MashZone NextGen Platform API Console" on page 1529 for more information.
- **Your Account**: to log out or use other utilities for your profile in MashZone NextGen. See "Your MashZone NextGen Hub Profile" on page 82 for more information.

Artifacts and Artifact Pages

Each mashable, mashup and app in MashZone NextGen has an artifact page that gives you access to run, edit, manage, publish, take snapshots and otherwise work with that artifact.



The tabs, menus and toolbars in artifact pages include:

- **Latest tab**: to preview or work with curren data for a mashable or mashup.
- **Run toolbar**: to run a mashable or mashup to preview results. For apps, preview is automatic. You can also:
  - Choose the current view to display results.
  - Take a snapshot of current data.
  - Schedule snapshots.

  For more information, see "Run and Preview Mashable/Mashup Data" on page 177, "Take, View or Delete Snapshots" on page 183 or "Schedule Snapshots" on page 185.
- **Collaboration toolbar**: to rate, share or comment on an artifact. You can also use this toolbar to mark an artifact as a favorite.

  For more information, see "Common Tasks for Mashables, Mashups and Apps" on page 85.
- **Show menu**: to view, and in some cases update general information, dependencies or technical details for the artifact. For more information, see:
  - "Common Tasks for Mashables, Mashups and Apps" on page 85 for basic task such as viewing or editing general information or viewing dependencies.
  -
  - "Use Mashable/Mashup Technical Specs" on page 1489 to view API connection information.
  - "Update Mashable Endpoints" on page 180
- **Create**: to:
  - Create a new mashup based on this mashable or mashup.

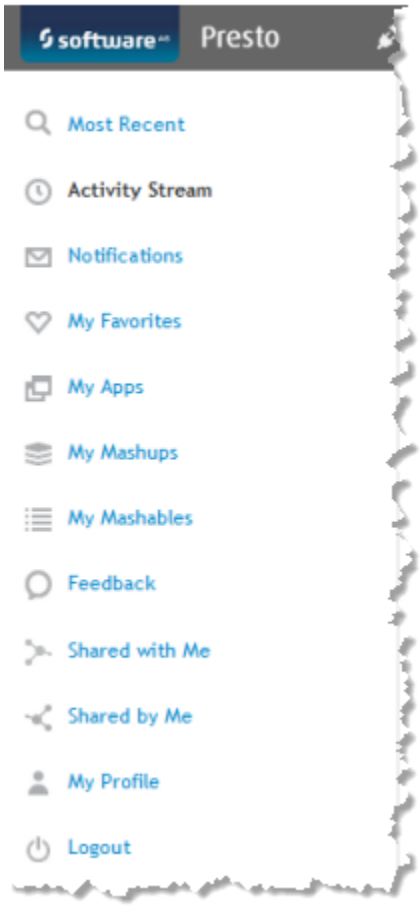
- "Create a Basic App" on page 1093 based on this mashable or mashup using the App Maker wizard.

- **Edit**: to edit mashups or apps in Wires, the App Maker wizard or Mashboard. Developers can also edit the mashup or app code using the Mashup Editor or App Editor.
- **View menu**: to add views, set the current view as the default, edit the current view or delete the current view.

  For more information, see "Add Views to Mashables and Mashups" on page 901 and "Manage Views" on page 904.
- **Publish menu**: to publish apps to other destinations such as the AppDepot or SharePoint. See "Publishing, Managing, Sharing and Using Apps" on page 1101 for more information.
- **Manage menu**: for owners and MashZone NextGen administrators to manage the artifact including granting run permissions to other users, changing the status of the artifact or deleting it. See these topics for more information:
  - "Common Tasks for Mashables, Mashups and Apps" on page 85
  - "Grant Permission to Run Mashables, Mashups and Apps" on page 102
  - "Update Mashable Security Profiles" on page 181
- **Snapshots tab**: where you can find and manage snapshots for mashables or mashups. See "Take, View or Delete Snapshots" on page 183 for more information.

The Home Page

In addition to the main menu, the Home page has short-cuts to your favorites, notifications that have been sent to you and recent activity by other users in MashZone NextGen Hub.



Your MashZone NextGen Hub Profile

Your user profile shows basic information about your account in MashZone NextGen and allows you to:

- Manage Your Locale and Account Information, if permitted
- Reset Your Password, if permitted
- Manage Your MashZone NextGen User Attributes

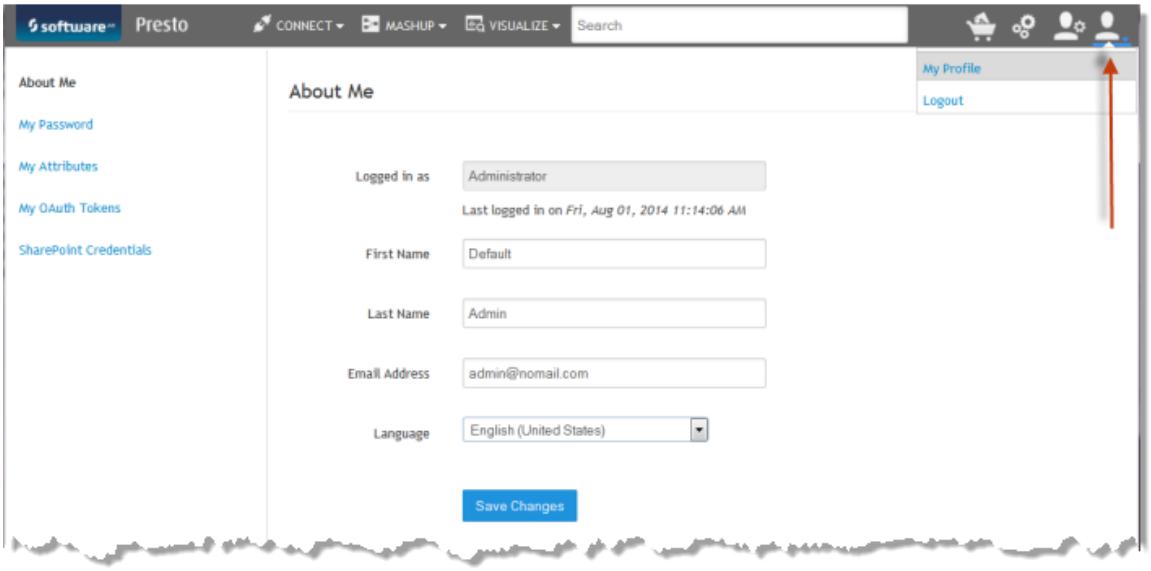
Manage Your MashZone NextGen User Attributes

User attributes generally store private information to use when you work with mashables, mashups or apps. For example, you may need to provide your email or a username to work with some mashables information sources. Mashables or mashups that use information from SharePoint may also need to use your credentials.

Note: Your MashZone NextGen administrator may also create global MashZone NextGen attributes to provide default information. If you don't want to use these defaults, you need to add a MashZone NextGen user attribute to your profile with the same name and supply the specific information you want to use.

To add a user attribute to your profile

1. Click **Your Account > Profile** from the MashZone NextGen Hub main menu.



2. Click **My Attributes**.
3. Enter the name for the attribute as the **Key** and enter the value that you want to use for this attribute.
4. Click **Save changes**.

You can also **Edit** or **Delete** any existing user attributes from the list.

Manage Your Locale and Account Information

Open your profile from the MashZone NextGen Hub menu bar and click **About Me** to see the basic information for your account in MashZone NextGen.

Change the **Language** preference, if needed and click **Save changes**.

In most cases, you cannot update any other account information because this comes from account information for your entire organization. In development or test environments where your account information is stored in the default MashZone NextGen Repository, you can save changes to this information.

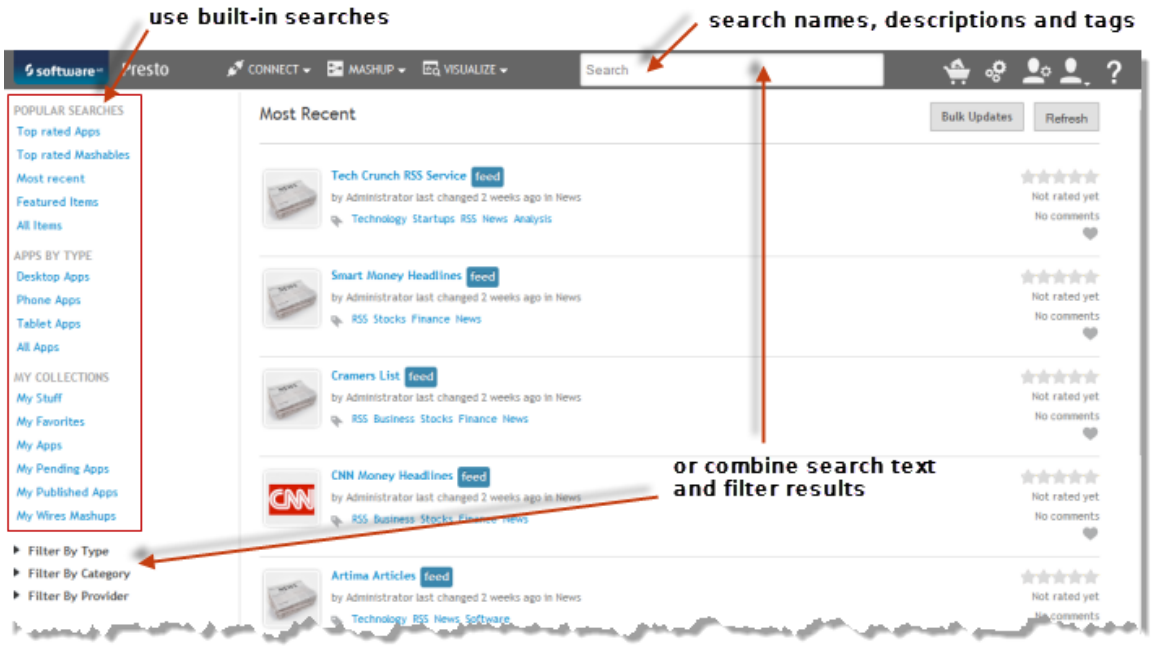
Reset Your Password

In most cases, you *cannot* update your password in MashZone NextGen because this comes from account information for your entire organization. In development or test environments where your account information is stored in the default MashZone NextGen Repository, however, you can reset your password from your profile.

Open your profile from the MashZone NextGen Hub menu bar and click **My Password**. Enter your new password and confirm this. Then click **Change Password**.

Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub

The most basic ways to find mashables, mashups or apps is with **Search** or your Favorites list:



You can also use filters with search text to shorten search results and save filtered searches as a bookmark.

Search results shows a summary for each artifact which you can use to open the artifact:

Click the artifact name to open its *artifact page* and work with that mashable, mashup or app. For more information, see:

- "Running Mashables or Mashups and Other Tasks" on page 176
- "Publishing, Managing, Sharing and Using Apps" on page 1101

You can also use **Bulk Updates** to select multiple artifacts that you want to update. See "Update Mashables, Mashups or Apps in Bulk" on page 97 for instructions.

Common Tasks for Mashables, Mashups and Apps

You work with mashables, mashup and apps individually from their artifact page. See "Artifacts and Artifact Pages" on page 79 for an overview.

Note: Some actions in mashable, mashup or app artifact pages may not be accessible based on your MashZone NextGen permissions.

Common tasks you can perform for artifacts include:

"Add Feedback" on page 86

"Add to Favorites or Bookmark" on page 94

"Add or Update Meta Data: Description, Category, Provider and Tags" on page 87

"Add or Update Custom Attributes" on page 88

"Add a Thumbnail" on page 89

"Change the Artifact Name" on page 91

"Change the Status" on page 93

"Delete" on page 95

"Export Results to CSV" on page 96

"Feature or Unfeature" on page 93

"Review Dependencies" on page 95

"Share With Other Users" on page 95

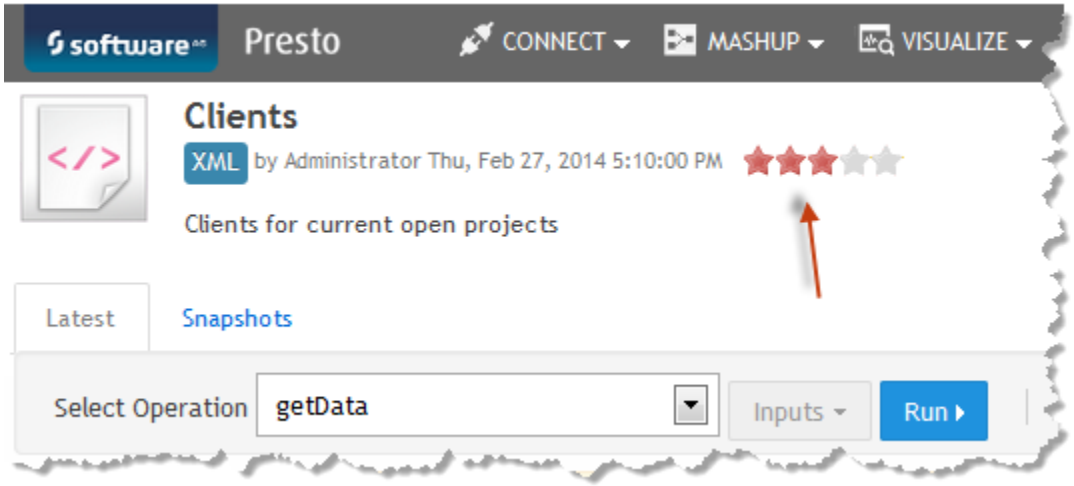
"Change an Artifact's Locale" on page 92

You can also perform some of these tasks to several artifacts at once, using the Bulk Update feature. See "Update Mashables, Mashups or Apps in Bulk" on page 97 for more information.

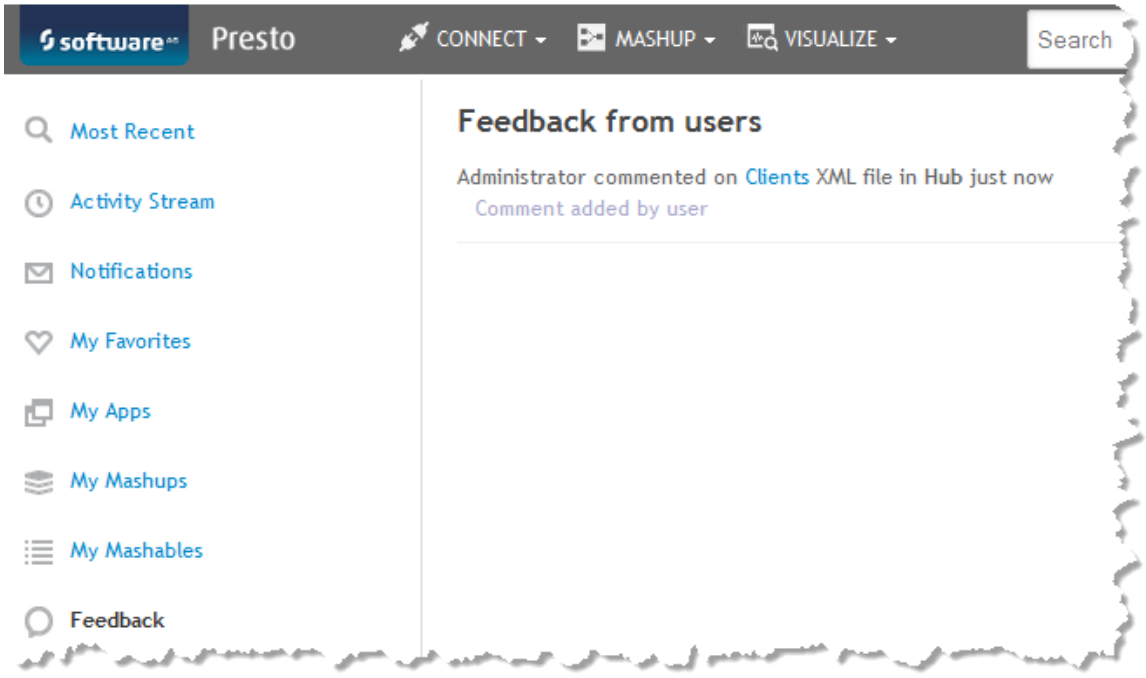
Add Feedback

Feedback includes rating and comments for mashables, mashups or apps in both MashZone NextGen Hub and the AppDepot. For published apps in the AppDepot, you can also send feedback to the app *owner* who created the app.

In MashZone NextGen Hub, use the toolbar in the artifact page to rate an artifact:



If you add comments in MashZone NextGen Hub, the artifact owner receives a notice of your comments in the MashZone NextGen Hub Home page.



Other users can also see your comments from the artifact page. To view comments or add a comment in MashZone NextGen Hub, open the artifact's page from search, bookmarks or other links and select **Show > Comments**.

In the AppDepot, you can rate or add comments to a published app or send feedback to the published app's owner using the toolbar in the Open Apps tab:

- **Rate or Add Comments** to a published app. These comments are only visible in the AppDepot.
- **Send Feedback** to the owner of a published app. The owner receives a notice of your comments and can see them in MashZone NextGen Hub. These comments are only visible in MashZone NextGen Hub.

Add or Update Meta Data: Description, Category, Provider and Tags

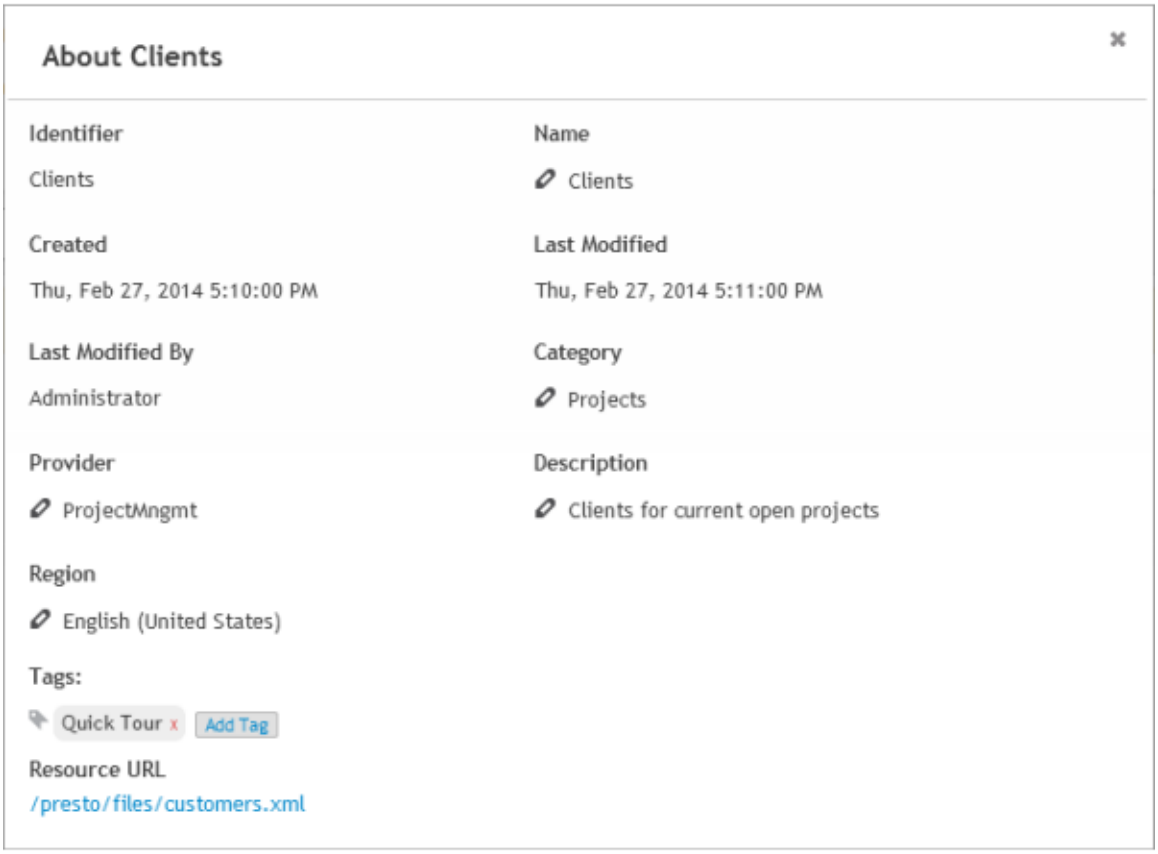
Depending on your MashZone NextGen permissions, you can add or update information about mashables, mashups and apps in MashZone NextGen Hub including the description, category, provider, region and tags.

Note: Published apps in the AppDepot should already have most of their meta data before they are published. You can, however, add tags to a published app in the AppDepot.

Category is the primary level of "what" this artifact does or what it is about. Tags refine that "what" and make it easier for you and other users to quickly find artifacts. The provider identifies "who" provides the information from this artifact.

To add or update meta data for an individual artifact in MashZone NextGen Hub

1. Open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links and select **Show >** **Info**:



2. To add or update the Category or Provider, click that field, select the appropriate category or provider and click **Save**.

> **Note:** If there is no suitable category or provider, contact your MashZone NextGen administrator to have them add one to these lists.

3. To update the Description, click the field, enter or update the description and click **Save**.

4. To add a tag, click ✚, enter the tag in the field that opens or select a tag from the list and click **OK**.

Or see "Update the Category, Provider or Tags in Bulk" on page 99.

Add or Update Custom Attributes

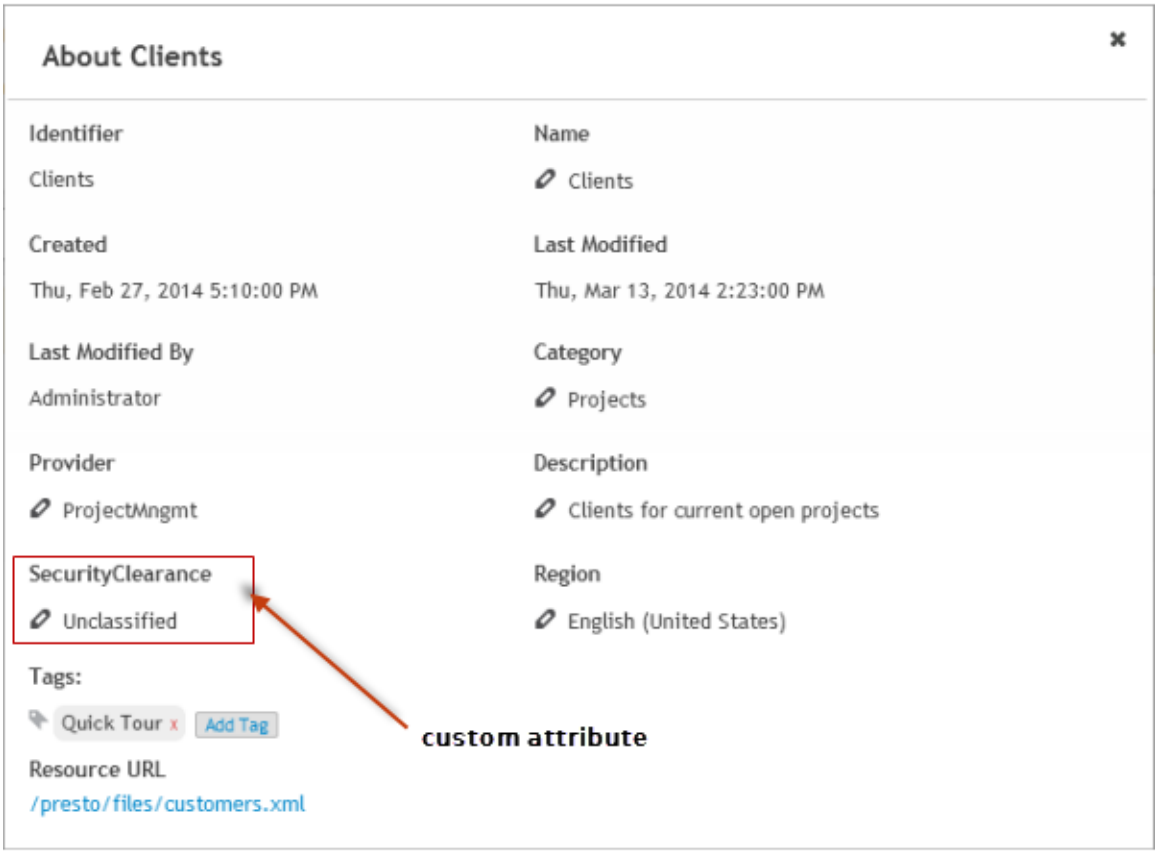
Your MashZone NextGen administrator can add *custom attributes* for apps, mashables or mashups to track additional information that is useful to your organization. For

example, the security clearance for a mashup could be used to determine who has permission to run that mashup.

If you have custom attributes for artifacts, you set values for these attributes in the **Info** tab for the artifacts that you create.

To set custom attribute values

1. Open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links and select **Show >** **Info**.
2. Find the custom attribute:



3. Click **Edit**.
4. Enter the appropriate value or select one from the list of values provided. Click **Save**.

Add a Thumbnail

MashZone NextGen provides a default thumbnail image for mashables, mashups and apps based on the type of artifact. You can replace this default image with a small image or screen capture that is more meaningful to other users.

Tip: Thumbnails appear in a space in MashZone NextGen Hub and in the AppDepot that is 72 pixels (roughly one inch) square. Screen captures that you use as thumbnails are reduced to fit into that space. To ensure better viewing quality, it is a good practice to use images that are close to this display size.

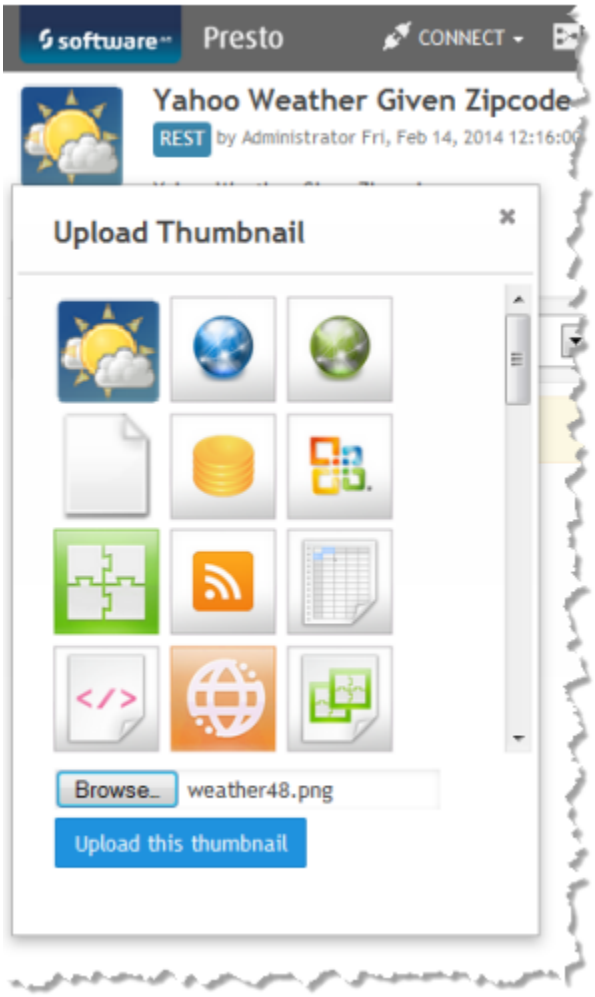
To add a thumbnail

1. If needed, capture an image of a screen, find a free icon, or find an image in your organization that you want to use as a thumbnail.

   Note: In Windows, you must paste screen captures from the clip board and save them as a file to use them as a thumbnail in MashZone NextGen.

2. Open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links.
3. Click the **Change** link that appears when you move the mouse over the thumbnail in the artifact page toolbar.

   This opens a window where you can select thumbnails that have already been uploaded or you can upload your image to use as the thumbnail.



4. Click on one of the existing thumbnails to use that image. To upload and use another image:

   a. Click **Browse**.

   b. Find the image file you want to upload and click **Open**.

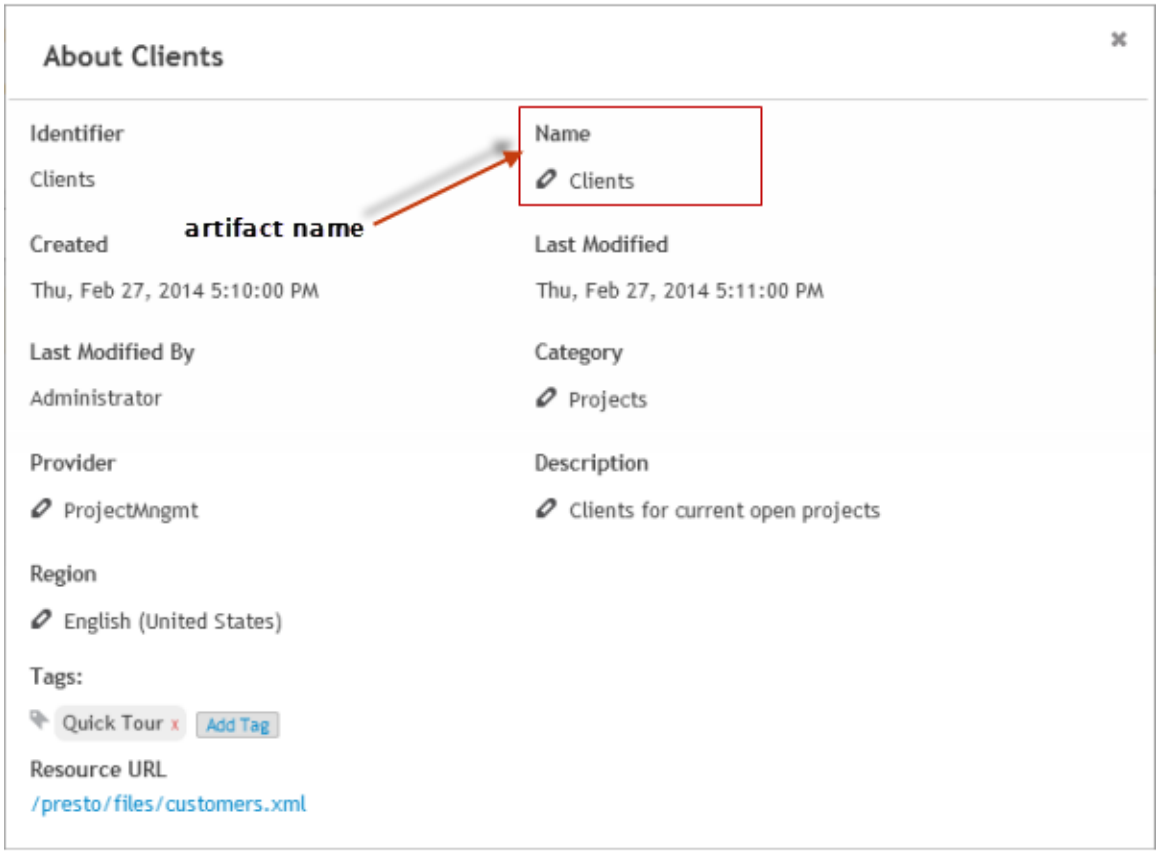
   c. Click **Upload this thumbnail**.

   This adds the image as a thumbnail to this artifact and also adds the image as a thumbnail that other users can select for other artifacts.

Change the Artifact Name

Owners can change the name of mashables, mashups and apps in MashZone NextGen Hub that they registered or created. MashZone NextGen administrators can change the name of any artifact. This does *not* change the ID that MashZone NextGen assigns when you first add or create an artifact.

To change a name, open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links and select **Show > Info**



Click on the artifact name, change the value and click **Save**.

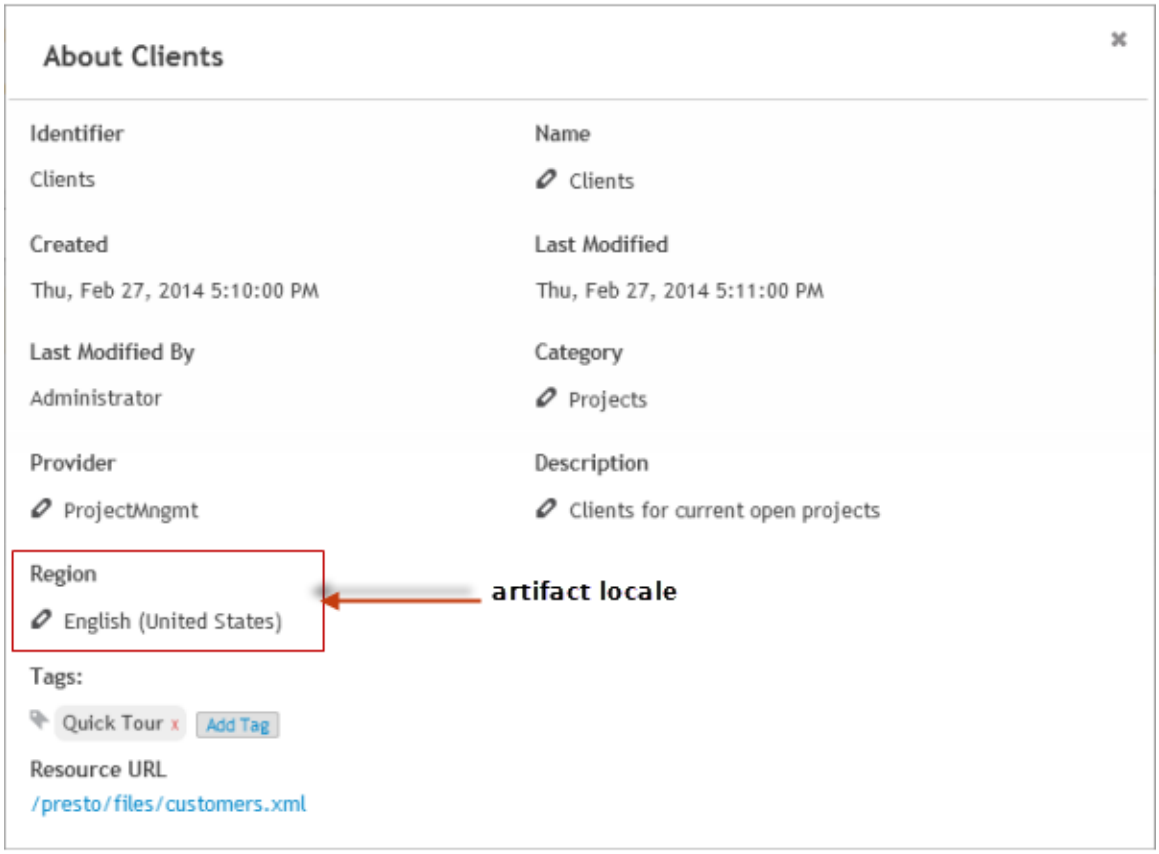
Change an Artifact's Locale

Locales define the language and formats used in a mashable's, mashup's or app's results for date, time and numeric data. The locale for artifacts defaults to either:

- The locale of the user who registered the mashable or created the mashup or app, if that was available.
- Or to American English (EN_us) if not.

If this default locale is incorrect for a mashable, dates, times and numbers in results may display incorrectly.

To change the locale, open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links and select **Show > Info**.



Click ✎ next to the **Region** field, select the locale and click **Save**.

Change the Status

You can turn mashables, mashups and apps off to temporarily stop other users from working with them while you update them and then turn them back on. Only artifact owners and MashZone NextGen administrators can change the status of artifacts.

To change the status for an individual artifact, open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links and select ⚙ **Manage >** ⏻ **Turn Off** or ⚙ **Manage >** ⏻ **Turn On** to change the status and confirm this choice.

Or see "Change the Status of Multiple Artifacts" on page 99.

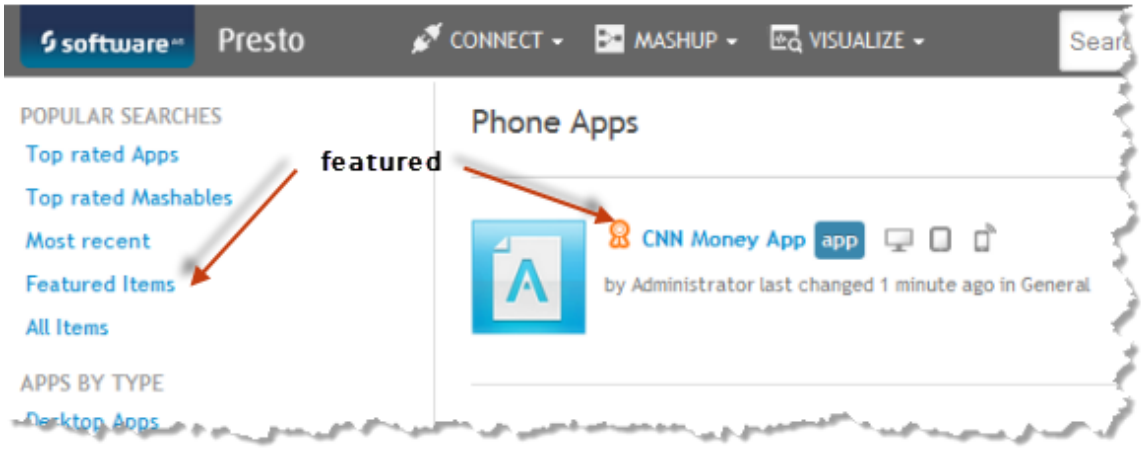
Feature or Unfeature

MashZone NextGen administrators can choose to feature specific mashables, mashups or apps to draw user attention to interesting artifacts. They can also remove these artifacts from featured lists in the MashZone NextGen Hub or in the AppDepot.

To feature an artifact in MashZone NextGen Hub

1. Open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links.
2. Select **Manage >** **Feature This**.

This artifact now appears in search results for the **Featured Items** in MashZone NextGen Hub.



To remove an artifact from MashZone NextGen Hub featured lists, open the artifact page in MashZone NextGen Hub, select **Manage >** **Unfeature This**.

To feature a published app in the AppDepot:

1. Find the app in the All Apps tab.
2. Open the app information page by clicking on the app name.
3. Click **Feature this App**.

To remove an app from the AppDepot Featured list, find the app, click the app's title and click **Unfeature this App**.

See also "Feature or Unfeature Multiple Artifacts" on page 101.

Add to Favorites or Bookmark

You can add any mashable information source, mashup or app to your Favorites list in MashZone NextGen Hub. Simply click from Search Results or from the artifact page toolbar.

Or see "Add To or Remove Multiple Artifacts From Favorites" on page 99.

You can also add any artifact to your Favorites or Bookmarks in your browser. Open the artifact page for that mashable, mashup or app and use your browser menus or toolbar to add the bookmark.

Delete

Owners and MashZone NextGen administrators can deletes artifacts permanently from MashZone NextGen. If other mashups or apps use this artifact, they are also affected. Before you remove an artifact, you should "Review Dependencies" on page 95.

To remove the artifact, select **Manage >** **Delete** and confirm this when prompted.

See also "Delete Multiple Artifacts" on page 100.

Review Dependencies

Before you remove an artifact permanently, you should review the artifacts that depend on it. Open the artifact in MashZone NextGen Hub and select **Show > Dependencies**.

This window lists both the dependencies (the artifacts that this mashup or app depends upon) and its dependents (the mashups or apps that use this artifact).

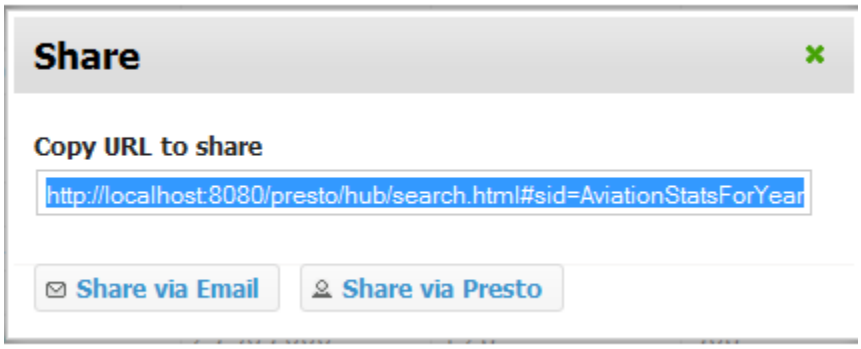
Share With Other Users

You can send notices within MashZone NextGen Hub to other MashZone NextGen users or send emails to any email address with a message and links to any mashable, mashup or app. You can also send sharing notices for published apps in the AppDepot.

Note: Your MashZone NextGen administrator must configure MashZone NextGen to support emails for sharing notices. You can always send sharing notices to other MashZone NextGen users.

> Sharing sends a notice, however, other users cannot use a shared artifact unless they have the proper permissions.

To share an artifact in MashZone NextGen Hub, open the artifact's page in MashZone NextGen Hub from search results, bookmarks, favorites or other links and click **Share**.



- Copy the URL and paste it in emails or documents.
- Click **Share via Email** to automatically start a new email in your default email application. A link to this artifact appears in the body of the email.
- Click **Share via MashZone NextGen** to send a notification in MashZone NextGen to another user. Enter MashZone NextGen usernames, separated by commas. Add an optional message, if desired, and click **Send**.

You receive sharing notices from other MashZone NextGen users in the **Shared With Me** tab on the MashZone NextGen Hub Home page. Notices of the artifacts you have shared with other users are listed in the **Shared By Me** tab.

In the AppDepot, you can also share published apps with other AppDepot users or send emails. Use **Share** in the app toolbar from the Open Apps tab, My Apps tab or the app information page.

Export Results to CSV

You can export the results shown in an app as CSV (comma-separated values).

To export results

1. Open the app's artifact page in MashZone NextGen Hub from search results, bookmarks, favorites or other links.
2. Click **Open in New Window**.

   The app opens in a new tab in the browser or a new browser window.
3. Click ▼ in the app toolbar and select **Export to CSV**.

4. When prompted, save the results as a file or open this in an application that can handle CSV data.

   If you save this as a file, the default file name is generally the operation name for the mashable or mashup used by this app, such as getData.

MashZone NextGen developers can also export mashable or mashup results as CSV from the artifact's technical specification. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information.

Update Mashables, Mashups or Apps in Bulk

You can use the bulk update feature in MashZone NextGen Hub to select multiple artifacts and apply some types of updates to all them at once. See "Common Tasks for Mashables, Mashups and Apps" on page 85 for instructions on updating individual artifacts.

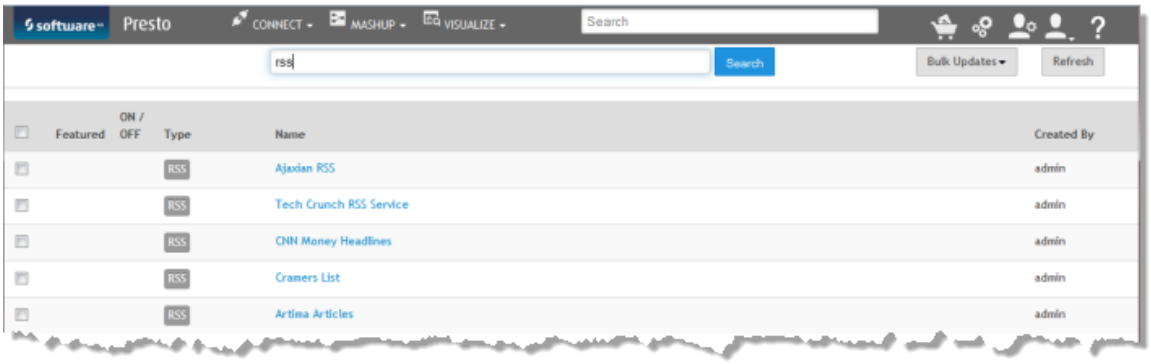
Note: The updates that you can perform depend on your MashZone NextGen permissions. Some tasks are limited to artifact owners or MashZone NextGen administrators.

First you Find and Select Multiple Artifacts for Update or Find and Select Artifact Dependencies. Then:

- Add To or Remove Multiple Artifacts From Favorites
- Update the Category, Provider or Tags in Bulk
- Change the Status of Multiple Artifacts
- Delete Multiple Artifacts
- Update Run Permissions for Multiple Artifacts
- Update Caching for Multiple Mashables or Mashups
- Feature or Unfeature Multiple Artifacts

Find and Select Multiple Artifacts for Update

1. Put the cursor in **Search**, enter search terms if desired and press **Return** to open the Search Results page.
2. Click **Bulk Updates**. The Bulk Updates page opens with a Search field.
3. Enter part of the name, description or a tag to get a list of artifacts and click **Search**. Or simply click **Search** to select all artifacts.



4. Set the check box next to each of the artifacts that you want to update. Or set the check box in the column headings to select all the artifacts in this list.
5. Complete one of these updates:
   - Add To or Remove Multiple Artifacts From Favorites
   - Update the Category, Provider or Tags in Bulk
   - Change the Status of Multiple Artifacts
   - Delete Multiple Artifacts
   - Update Run Permissions for Multiple Artifacts
   - Update Caching for Multiple Mashables or Mashups
   - Feature or Unfeature Multiple Artifacts

Find and Select Artifact Dependencies

Another common set of artifacts you may need to update are dependencies, the mashables, mashups or apps used by an artifact. To select dependencies:

1. Open the app, mashup or mashable artifact page from search results, favorites or bookmarks.
2. Select **Manage >** **Dependencies**.

   This opens the Bulk Updates page with the list of all dependencies for this mashup or app.
3. Set the option next to each of the dependent artifacts that you want to update. Or set the option in the column headings to select all of the dependencies.
4. Complete one of these updates:
   - Add To or Remove Multiple Artifacts From Favorites
   - Update the Category, Provider or Tags in Bulk
   - Change the Status of Multiple Artifacts

- Delete Multiple Artifacts
- Update Run Permissions for Multiple Artifacts
- Update Caching for Multiple Mashables or Mashups
- Feature or Unfeature Multiple Artifacts

Add To or Remove Multiple Artifacts From Favorites

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates** and select either:
   - **Add to favorites**
   - **Remove from favorites**

Update the Category, Provider or Tags in Bulk

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates**.
3. Select **Edit Info** from the menu.
4. Set the check box for the fields you want to update.
5. Then select a Provider or Category or enter one or more Tags.

   Important: If you choose to update tags, the list of tags that you provide replaces any existing tags for these artifacts.

6. Click **Save changes**.

With all of these fields, the value(s) you select or enter replace the existing Provider, Category or Tags.

Change the Status of Multiple Artifacts

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates** and select either:
   - **Turn ON**
   - **Turn OFF**

The new status for the selected artifacts displays in the Bulk Updates list.

Delete Multiple Artifacts

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates**.
3. Select **Delete** in the menu and confirm this when prompted.

Update Run Permissions for Multiple Artifacts

For mashable artifacts that have multiple operations, you can only update permissions for the whole mashable (all operations) with bulk updates. If you need to add permissions for specific operations, you must update each mashable individually. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates**.
3. Select **Permissions** from the menu.
4. In the Update Permissions window:
   a. Set options to select groups or users or both.
   b. Enter part of a name to find the groups or users you want to grant run permissions to and click **Search**
   c. Click each group or user in the **Search Result** list that you want to grant run permissions to.

      The groups and users you select move to the **Selected Principals** list.
   d. Once the Selected Principals list has all the groups or users to want to grant run permission to, click **Assign Permissions**.

The groups and users you have selected are added to the list of principals with run permissions for all the selected artifacts.

Update Caching for Multiple Mashables or Mashups

For mashable artifacts that have multiple operations, you can only update caching for the whole mashable (all operations) with bulk updates. If you need to set caching for specific operations, you must update each mashable individually.

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates**.
3. Select **Caching** from the menu.
4. Set the **Enable Caching** property to either:

- `Inherit global settings` = caching is controlled by configuration set by MashZone NextGen administrators for all mashables of this type or for all mashups.
- `Yes` = turn caching on for these artifacts regardless of global caching configuration.
- `No` = turn caching off for these artifacts regardless of global caching configuration.

5. If you turned caching on or inherited the global settings, set the **Max Cache Age** property to either:
   - `Inherit global settings` = use the default maximum cache entry age set by MashZone NextGen administrators for all mashables of this type or for all mashups.
   - Choose one of the specific time limits as the maximum age for cache entries for this mashable or mashup regardless of global caching configuration.
6. Click **Save Changes**.

Feature or Unfeature Multiple Artifacts

Only MashZone NextGen administrators may feature or unfeature artifacts.

1. Find and Select Multiple Artifacts for Update.
2. Once you have selected artifacts, click **Bulk Updates** and select either:
   - **Add to featured list**
   - **Remove from featured list**

The new featured status for the selected artifacts displays in the Bulk Updates list.

Descriptions, Providers, Categories and Tags for Artifacts

Mashables, mashups and apps in MashZone NextGen all have *metadata* which you can add to help you and other users find and recognize the artifacts. This includes Category, Description, Provider, and Tags.

Category

Categories identify the primary purpose, product, area of interest, aspect or other grouping for this artifact or resource. Categories define what an artifact pertains to.

Categories are defined by MashZone NextGen administrators.

Description

Descriptions are generally one or two short sentences that more fully describe the purpose, use or scope of an artifact.

Provider

Providers are the organization, department or group who provides or is responsible for the information in this artifact. Providers define who or where information comes from. This can be external sources or systems or groups within your own organization.

Providers are defined by MashZone NextGen administrators.

Tags

Tags are one subject, purpose or other aspect of an artifact. Tags define a finer grain of what an artifact is about. Artifacts can have any number of tags. Tags are defined by users.

Grant Permission to Run Mashables, Mashups and Apps

Other users cannot work with mashables that you register or mashups or apps that you create until you, or a MashZone NextGen administrator, grant them permission to run the artifact.

For mashables with several operations, you can grant permission to run all operations or to run individual operations.

For mashups and apps, you, or a MashZone NextGen administrator, can grant permission to run the mashup or app in MashZone NextGen Hub. Users must also have run permissions for any *dependencies*: the mashables, mashups or other apps used in this mashup or app.

When you publish apps to the AppDepot, these run permissions are also applied to the apps in the AppDepot. AppDepot Managers can, however, change the permissions to run the published app.

There are several ways to grant run permissions. You can:

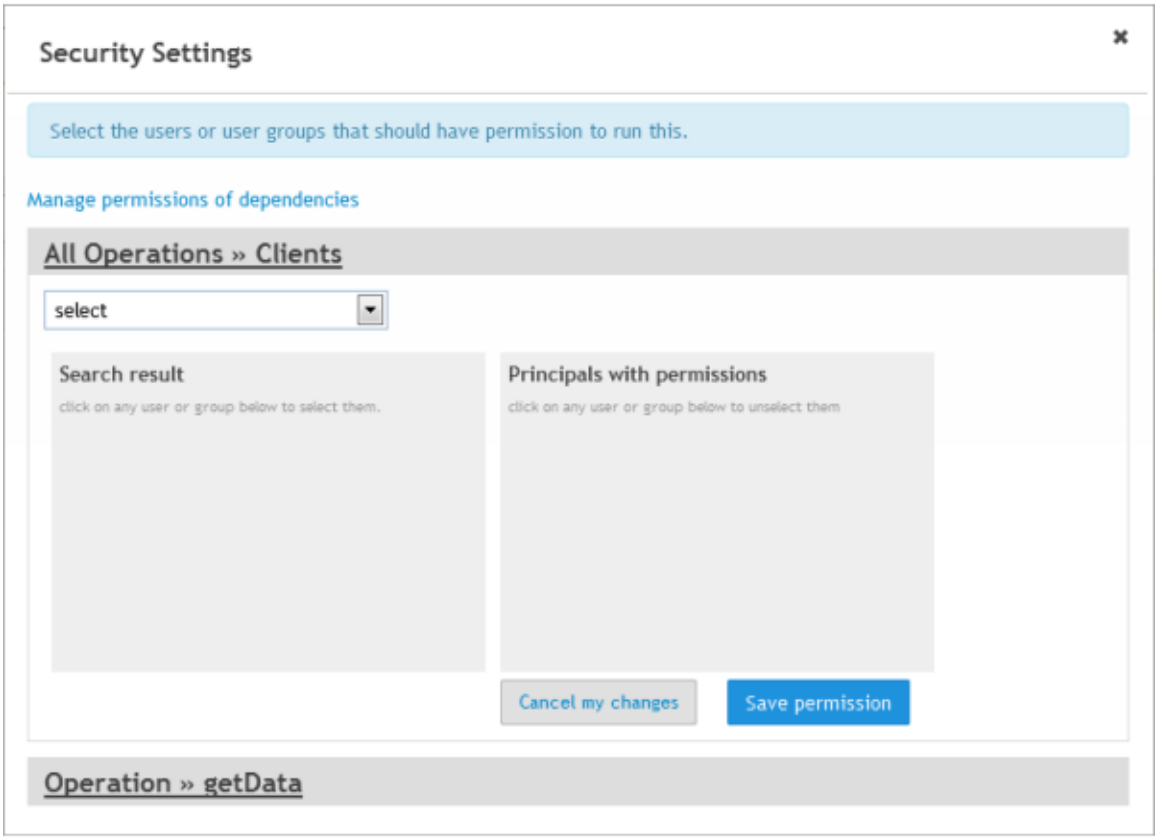
- Grant Run Permissions for One Artifact.

  If needed, you can also Grant Run Permissions to an Artifact's Dependencies.
- Grant run permissions to several artifacts you select. See Update Mashables, Mashups or Apps in Bulk for instructions.
- Grant run permissions to published apps. See Grant Run Permissions to Published Apps in the AppDepotfor instructions.

Grant Run Permissions for One Artifact

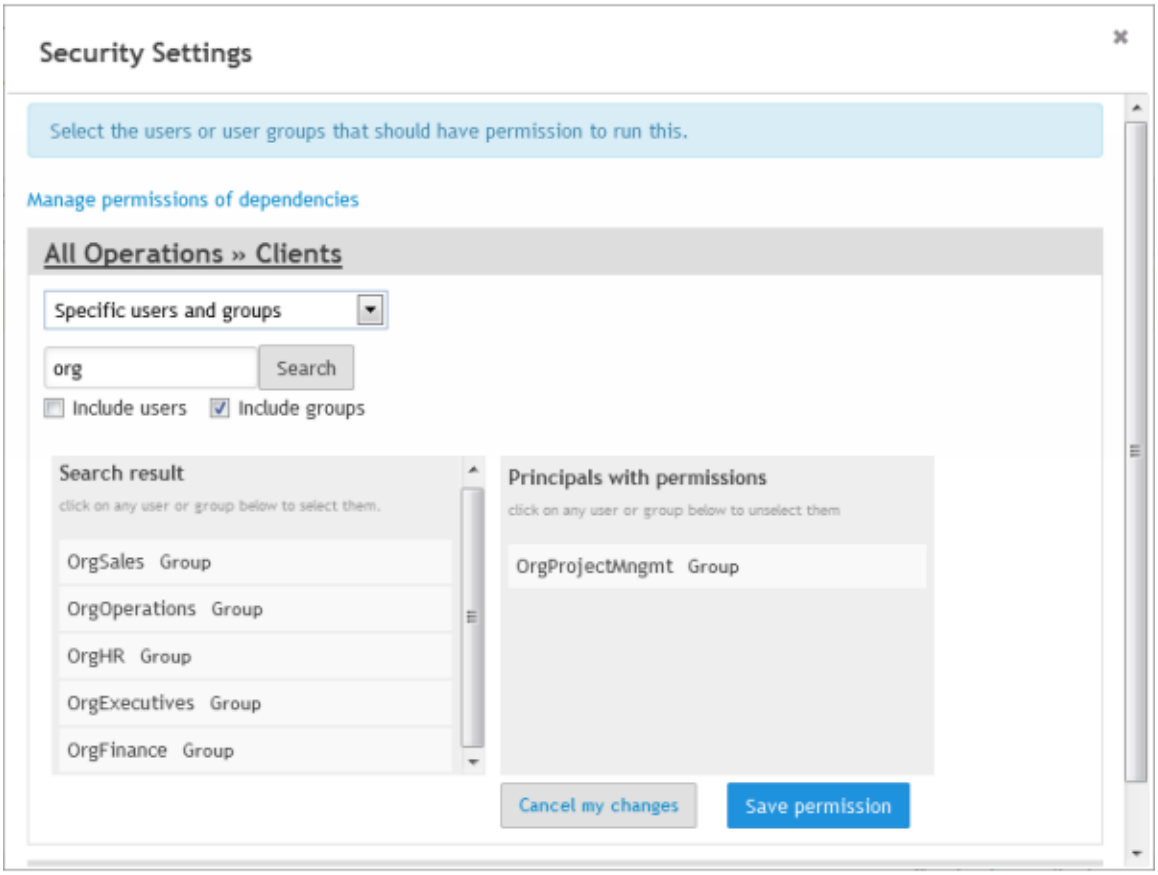
Run permissions allow other users to work with a mashable, mashup or app that you have created.

1. Open the app, mashup or mashable artifact page from search results, favorites or bookmarks.
2. Select **Manage >** **Permissions**



Some groups or users may already be listed as **Principals with permissions**. These users and groups have been automatically or manually granted permission by a MashZone NextGen administrator.

3. For mashables with multiple operations, choose the mashable-wide **Service >>** ***mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >>** ***operation name*** section to grant run permissions for just that operation.
4. Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
5. To grant run permissions, select a user or group in the **Search result** list to add them to the list of **Principals with permissions**.

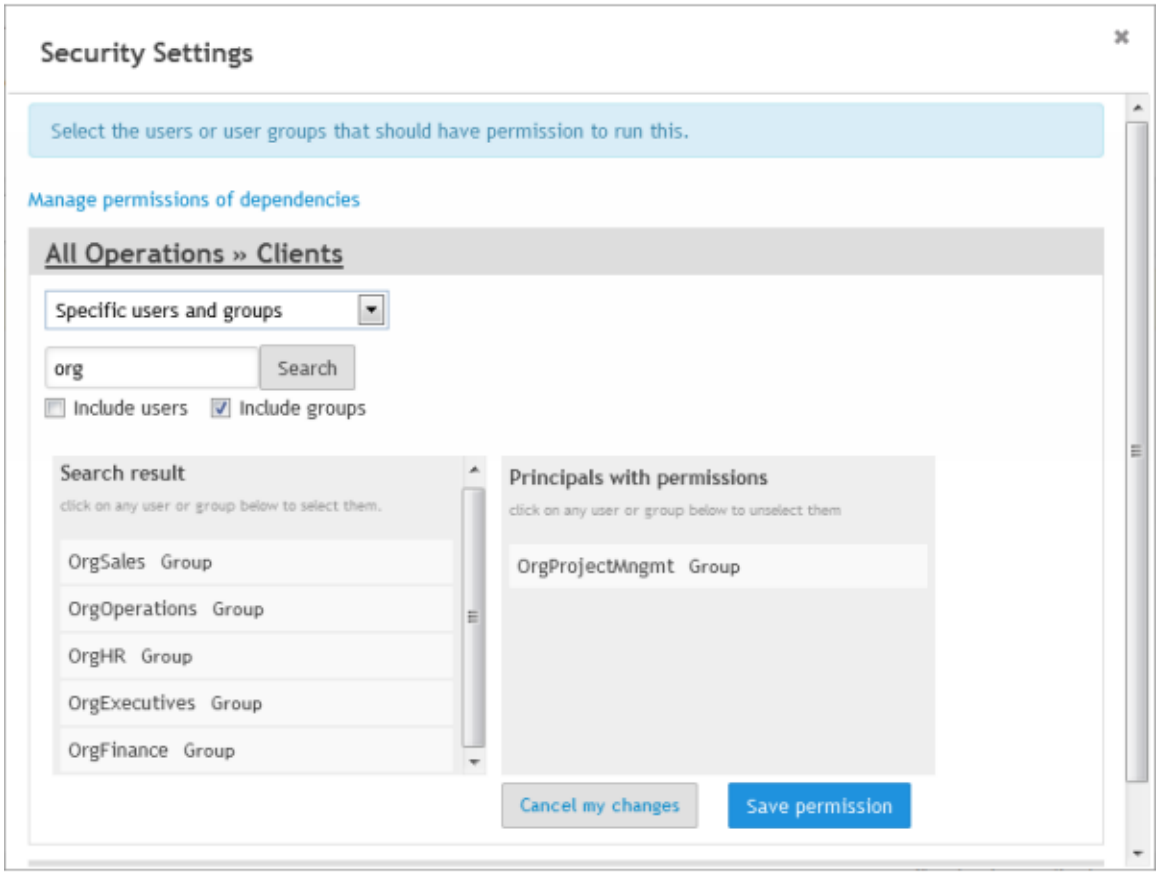


6. To revoke run permissions, click on a user or group in the **Principals with permissions** list.
7. Click **Save permission** and close the Security window.
8. If needed, you can also Grant Run Permissions to an Artifact's Dependencies.

   To use mashups and apps, other users must also have run permissions for the artifact's dependencies - any other artifacts that the mashup or app uses.

Find Users and Groups for Permissions

To find individual users or groups of users for your organization that you want to grant run permissions to, select `Allow specific users and groups` in Security Settings.



Then clear or set the **Search users** or **Search groups** options as needed. Enter part of a user or group name and click **Search**.

The **Search result** pane shows a list of users and groups that match your criteria.

Choose MashZone NextGen Built-In Groups for Permissions

You can also select any of the MashZone NextGen built-in groups to grant run permissions for an artifact:

- `Allow all authenticated and guest users` = anyone can run this app, mashup, mashable or mashable operation even guest users who have not logged in. This is also known as granting guest or anonymous access.
- `Allow all authenticated users` = any user who has logged in can run this app, mashup, mashable or mashable operation.
- `Allow all power users` = any user who is assigned to the built-in MashZone NextGen_PowerUser group can run this app, mashup, mashable or mashable operation. Typically, this group is for non-technical users who can work in MashZone NextGen Hub to create basic apps, create mashups in Wires and create workspace apps in Mashboard.

- `Allow all developers` = any user who is assigned to the built-in MashZone NextGen_Developer group can run this app, mashup, mashable or mashable operation. Typically, this is IT developers who work in MashZone NextGen Hub and can use both simple tools and code editors to create apps and mashups.
- `Allow all power user and developers` = any user who is assigned to the built-in MashZone NextGen_PowerUser or MashZone NextGen_Developer groups can run this app, mashup, mashable or mashable operation. This basically is anyone with access to MashZone NextGen Hub.

Selecting one of these groups automatically adds the group to the **Principals with permissions** list.

Grant Run Permissions to an Artifact's Dependencies

Run permissions allow other users to work with a mashable, mashup or app that you have created. For mashups and apps, users must also have run permissions for any dependencies: the mashables or other mashups and apps that are used by this artifact.

To grant or update run permissions for mashup or app dependencies

1. Open the app, mashup or mashable artifact page from search results, favorites or bookmarks.
2. Select **Manage >** **Dependencies**.

   This opens the Bulk Updates page with the list of all dependencies for this mashup or app.
3. Set the option next to each of the dependent artifacts that you want to update. Or set the option in the column headings to select all of the dependencies.
4. Once you have selected artifacts, click **Bulk Updates**.
5. Select **Permissions** from the menu.
6. In the Update Permissions window:
   a. Set options to select groups or users or both.
   b. Enter part of a name to find the groups or users you want to grant run permissions to and click **Search**
   c. Click each group or user in the **Search Result** list that you want to grant run permissions to.

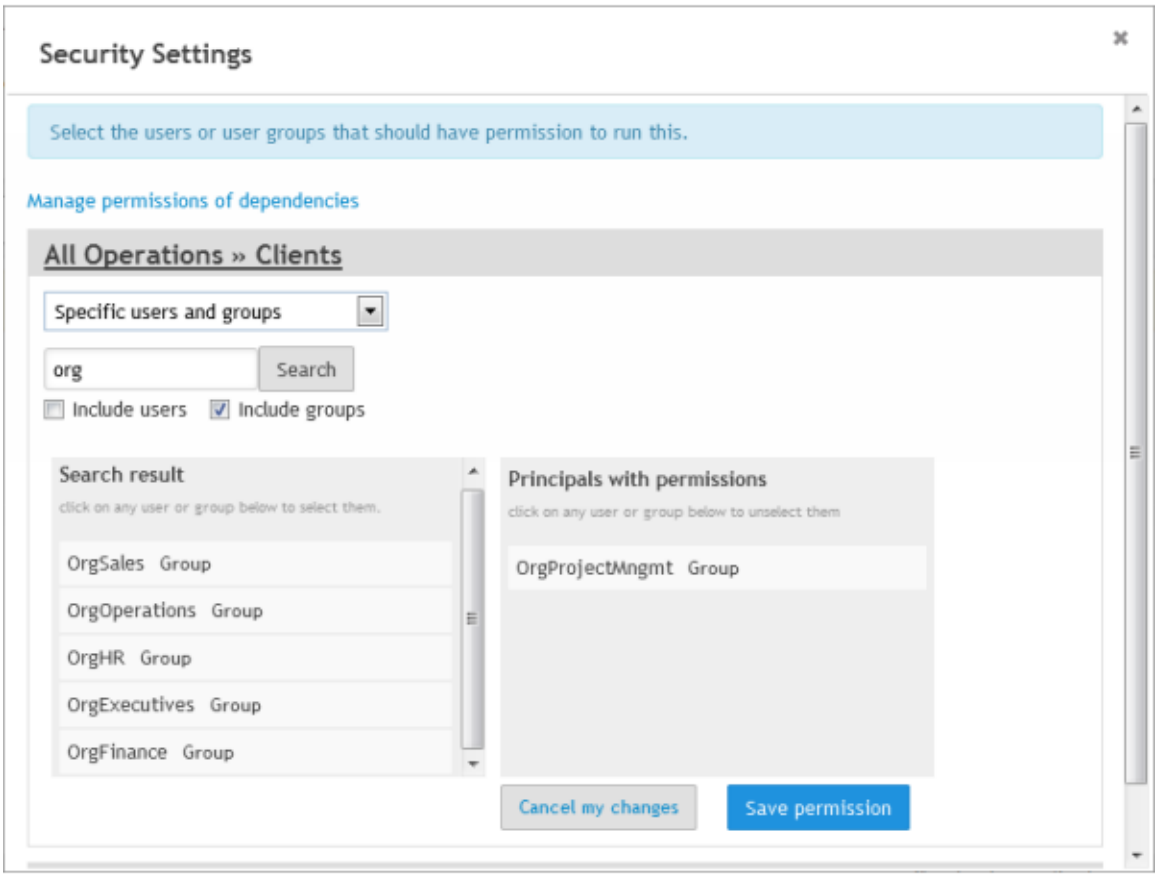
      The groups and users you select move to the **Selected Principals** list.
   d. Once the Selected Principals list has all the groups or users to want to grant run permission to, click **Assign Permissions**.

You can also use other functions on the Bulk Update menu to update artifact dependencies. For more information on other update options, see "Update Mashables, Mashups or Apps in Bulk" on page 97.

Grant Run Permissions to Published Apps in the AppDepot

Published apps in the AppDepot have the same run permissions as the app in MashZone NextGen Hub. Only AppDepot Managers can update run permissions for published apps.

1. If the app is not yet approved, review the app and approve it, if appropriate. See "Managing Pending Apps in the AppDepot" on page 1187 for instructions.
2. Find the app in All Apps and click a published app to open its Information page.
3. Click **App Security** and Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
4. To grant run permissions, select a user or group in the **Search result** list to add them to the list of **Principals with permissions**.



5. To revoke run permissions, click on a user or group in the **Principals with permissions** list.
6. Click **Save permission** and then close the Security window.

3 Mashables

- Types of Mashable Information Sources 110
- Connect Information Sources as Mashables 136
- Running Mashables or Mashups and Other Tasks 176

Mashable information sources are the fundamental data sources that you use to create mashups and apps. These information sources may be applications, databases or files that belong to your organization. They can also be *events* generated by business process tracking or other business intelligence processes. Or, they can be information sources that are publicly accessible.

MashZone NextGen supports a wide variety of information sources that you can register as *mashables*. This includes common standards such as web services, web feeds and databases, many common types of files, such as spreadsheets, or information from other Software AG applications such as *business events* or MashZone feeds. See "Types of Mashable Information Sources" on page 110 for a summary of MashZone NextGen's support and links to more information.

See "Connect Information Sources as Mashables" on page 136 for links to instructions on how to connect and register information sources, subscribe to events or work with MashZone data feeds.

For event and Apama mashables, only MashZone NextGen administrators can create these mashables.

You can use mashables in mashups to combine, transform or further refine the information you are interested in. See "Mashups in MashZone NextGen Wires" on page 281 and "Mashups in EMML" on page 441 for links to more information on creating mashups.

You can also use mashables directly as the source of information for any number of basic apps or custom apps. See "Apps and Workspaces" on page 1091 for links to more information on creating apps from mashables or mashups.

You can run mashables to play with the information, add or manage views to configure how the information can be formatted, manage configuration for mashables and many other common tasks, See "Running Mashables or Mashups and Other Tasks" on page 176 for links to working with mashable information sources.

Types of Mashable Information Sources

MashZone NextGen supports several different types of information sources to use as mashables. This includes well-known web feed and web service standards, common business applications or documents, such as databases or SharePoint lists, and MashZone feeds. It also includes business events from other Software AG applications. You can also use some files as mashable information sources (spreadsheets, CSV and XML) or snapshots taken from other mashables or mashups.

Note: You can also make some other information sources accessible in MashZone NextGen by creating mashups that retrieve their information. Common examples include *web clipping* to retrieve information directly from web pages or *POJO/Java* applications.

The types of information sources that MashZone NextGen supports for mashables include:

Information Source	Description	Mashable Type
Apama	Information from Apama is available as events through the Event Bus. (See Events below for more information.) MashZone NextGen also directly accesses the following types of Apama data: ■ *Scenarios* in Apama have event data transformed specifically for use in dashboards. (This is also sometimes called *dataviews*.) MashZone NextGen works with Apama scenario events through the Event Service and event sources defined by a MashZone NextGen administrator. See "Event Service Configuration and Administration" on page 1704 for more information. ■ *Distributed Stores* in the Apama MemoryStore hold data that can be shared across several correlators and monitors and can be used in dashboards. In some cases, this may be a very large dataset. Data from these distributed Apama stores cannot be accessed as scenarios or as events. If the Apama MemoryStore uses BigMemory, however, MashZone NextGen mashups can access this data using MashZone NextGen Analytics and *dynamic external stores*. See "Example 129. Load Datasets from a Dynamic External In-Memory Store" on page 872 for an example.	■ Event ■ Mashup
CSV Files	CSV files can be information sources exported from spreadsheets, some applications or some databases. MashZone NextGen keeps a snapshot of this information when you register the mashable. To update information, you must re-register the mashable.	REST

Information Source	Description	Mashable Type
	See "REST Web Services" on page 129, "Register CSV Mashables" on page 146 and "Valid Labels and Values for Spreadsheet/ CSV Mashables" on page 131 for more information.	
Events for Business or Other Processes	Events are most commonly real-time status information about business processes that are monitored in your organization. Events are published by various Software AG applications to the Event Bus which then routes these events to applications, such as MashZone NextGen, who have subscribed to events of that type. Event mashables are subscriptions to a specific event type that make event data available in MashZone NextGen for use in apps. Unlike other types of mashables, however, you must have MashZone NextGen administrator permissions to create event mashables.	Event
Excel Spreadsheets	Spreadsheet mashables can be snapshots of data from a spreadsheet or a dynamic link to a spreadsheet. They can access data from a range of cells in one worksheet. You can create spreadsheet mashables in Excel using the MashZone NextGen Add-On for Excel or register them in MashZone NextGen Hub. For more information, see: ■ "Spreadsheet Mashables" on page 130 ■ "Valid Labels and Values for Spreadsheet/CSV Mashables" on page 131 ■ "Register an Excel Worksheet File from MashZone NextGen Hub" on page 143	Spreadsheet
MashZone Feeds	Data sources added and transformed in MashZone data feeds. These feeds are similar to mashups, but were created in MashZone and imported to MashZone NextGen. Source information can come from files or databases much like those supported by MashZone NextGen. Information can also	DataFeed

Information Source	Description	Mashable Type
	come from business or other types of events published by other Software AG applications such as Optimize. See "MashZone Feeds" on page 128 for more information.	
Relational Databases	Tables, views and stored procedures in relational databases can be exposed as mashable information sources. MashZone NextGen supports any database with a JDBC 2.1 compliant driver. See "Database Mashables" on page 115 for more information.	Database
REST Web Services	REST web services can include public web services or applications in your organization that are accessible by URL using either the GET or POST HTTP methods. See "REST Web Services" on page 129 for more information.	REST
SharePoint Lists	To use SharePoint lists as a mashable information source, you must have the MashZone NextGen Add-On for SharePoint installed and configured in MashZone NextGen.	SharePoint
Snapshots of other Mashables or Mashups	You can register individual snapshots that you have taken of any mashable or mashup as a mashable information source. This allow you to add views and create basic apps from snapshots or quickly share snapshots with other users. See "Snapshot Mashables" on page 130 and "Register a Snapshot as a Mashable" on page 158 for more information.	Mashup
Web Feeds	You can register web feeds for a variety of syndication feed formats, including: ■ RSS 0.90 ■ RSS 0.91 Netscape	RSS or Atom

Information Source	Description	Mashable Type
	■ RSS 0.91 Userland ■ RSS 0.92 ■ RSS 0.93 ■ RSS 0.94 ■ RSS 1.0 ■ RSS 2.0 ■ Atom 0.3 ■ Atom 1.0 **Note:** MashZone NextGen now supports both Atom and RSS feeds by default. However, feed results are *not* normalized to a single format. See "Syndicated Feeds (RSS/Atom)" on page 131 for more information.	
WSDL Web Services	These include web services that use SOAP versions 1.1 or 1.2 in document style or RPC style. Services must comply with the WS-I Basic profile. SOAP web services use WSDLs (web service definition language) to define the service contract. See "WSDL Web Services" on page 133 for more information about registering and working with WSDL services.	WSDL
XML Files	You can use any well-formed XML file as a mashable information source. MashZone NextGen keeps a snapshot of this information when you register the mashable. To update information, you must re-register the mashable. See "Register XML Mashables" on page 145 for more information.	■ XML, for MashZone NextGen 3.5 or later ■ REST for MashZone NextGen 3.2.1 or earlier

Database Mashables

Database mashables in MashZone NextGen allow you to work with the information in tables, views or stored procedures in your databases. You choose to expose database assets in mashables when you register them in MashZone NextGen.

Note: You can also work with database information in mashups using a built-in block in Wires (see "Run a SQL Statement" on page 348) or various statements in EMML (see "Issuing SQL Statements Directly to a Datasource" on page 491).

You can register simple database mashables that work with a single table, view or stored procedure. See "Register Basic Database Mashables" on page 147 for instructions. MashZone NextGen administrators can also create much more complex database mashables with multiple database artifacts and custom queries. See "Register Custom Database Mashables" on page 149 for instructions.

See any of the following topics for additional information on database mashables:

- "Database Mashable Names versus Database Names" on page 116
- "Default Operations Available for Database Mashables" on page 118
- "Arbitrary SQL Queries for Database Mashables" on page 124
- "Database Mashables Responses" on page 125
- "Database Mashable Support and Limitations" on page 115

Database Mashable Support and Limitations

MashZone NextGen supports databases with a JDBC 2.1 compliant driver. Some databases support valid names that MashZone NextGen does not support. See "Database Characters and Names That Are Not Supported" on page 117 for details.

Database mashables can use any combination of tables, views or stored procedures defined in one schema or catalog within one datasource. MashZone NextGen does not support distributed transactions.

MashZone NextGen does not support columns with an image datatype.

Currently, MashZone NextGen handles packages for stored procedures and REF CURSOR types in Oracle databases only, with the following limitations:

- Both weakly- and strongly-typed REF CURSORs are supported.
- REF CURSORs are supported only in OUT parameters for stored procedures or as the return type for functions.
- REF CURSORs are not supported as IN or INOUT parameters.
- Nested REF CURSORs are not supported.

See also "Valid Date Formats for MashZone NextGen Mashables and Mashups" on page 179 for the specific date formats you may use in input parameters to database mashable operations.

EEEE, MMMM DD, YYYY	MMMM DD, YYYY	MMM DD, YYYY

Where *EEEE* is the day of the week, *MMMM* is the full month name and *MMM* is a three-character abbreviation of the month.

MM/DD/YY hh:mm:ss a z	Or these formats where time uses a twelve-hour clock (1-12), *a* is AM or PM and *z* indicates the time zone.
MM/DD/YY hh:mm:ss a	
MM/DD/YY hh:mm a	
MM/DD/YY	

And finally, these formats where time uses a 24-hour clock

YYYY/MM/DD hh:mm:ss.sss	YYYY-MM-DD hh:mm:ss.sss	YYYY.MM.DD hh:mm:ss.sss
YYYY/MM/DD hh:mm:ss	YYYY-MM-DD hh:mm:ss	YYYY.MM.DD hh:mm:ss
YYYY/MM/DD hh:mm	YYYY-MM-DD hh:mm	YYYY.MM.DD hh:mm
YYYY/MM/DD	YYYY-MM-DD	YYYY.MM.DD

Database Mashable Names versus Database Names

MashZone NextGen database mashables use Java naming conventions. Mashable names are derived from the SQL names from the database for tables, views, stored procedures and columns. See "Database Mashable Name Changes" on page 116 for information on how mashable names differ from the corresponding database names.

Valid database names may not be valid in MashZone NextGen. See "Database Characters and Names That Are Not Supported" on page 117 for information on support.

Database Mashable Name Changes

The changes to names in database mashables include:

- Table, view and column names in the mashable use mixed cases, commonly called *camel case*. Table and view names start with an initial capital letter and capitalize the initial letter of all subsequent words. All other characters become lowercase.

  The first word of column names is all lowercase and subsequent words use an initial capital letter with lowercase.

- Periods used between packages and stored procedure names, in Oracle databases, are removed.

- Underscores (_) and spaces are removed from all names.
- The letter `N` is prepended to table, view or stored procedure names that begin with numbers. The letter `n` is prepended to column names that begin with numbers.
- The letter `a` is prepended to column names that are reserved words in Java.
- Operation names for stored procedures are in the form `execute`*Stored-procedure-name-in-camel-case* .
- If any removal or substitution results in duplicate names in a mashable, an index number is added to the new name to ensure uniqueness.
- Overloaded names, such as stored procedure names or operations with multiple signatures, have an index number added to ensure uniqueness.

For example:

For	Database SQL Name	Mashable Name
Tables or Views	EMPLOYEES	Employees
	lowercasename	LowercaseName
	AUDIT_LOG	AuditLog
	15CASES	N15Cases
Columns	JOB_ID	jobId
	3ANSWERS	n3Answers
	class	aClass
Stored Procedures	abc.SomeProcedure	executeAbcSomeProcedure
	AnotherProcedure(string) AnotherProcedure(string, string) AnotherProcedure(string, number)	executeAnotherProcedure executeAnotherProcedure0 executeAnotherProcedure1

Database Characters and Names That Are Not Supported

When you register database mashables, MashZone NextGen can detect and automatically handle some problems caused by database names that use unsupported

characters or reserved words. MashZone NextGen cannot detect or handle all invalid names which can cause registering to fail for a database mashable.

In general, the following characters or reserved words in database names may cause errors when you attempt to register a database mashable:

- Period (.) or colon (:).
- Dollar sign ($), percent (%) or common wildcard characters (* and ?).
- Common grouping punctuation, such as parentheses, braces or brackets ((), {} or []).
- XML delimiters such as <, > or &.
- Database names using SQL reserved words. For example, names such as `SCHEMA` are not supported.

If registering fails because of invalid names, you can reregister the mashable removing the offending column, table or view during registering. Or, you can reregister and edit configuration. Enclose the offending **SQL Name** value in quote marks (").

Default Operations Available for Database Mashables

MashZone NextGen creates the following operations in database mashables by default. With custom database mashables, MashZone NextGen adminsitrators can remove any of these default operations when they register database assets in MashZone NextGen. With simple database mashables, all default operations are generated.

Note: MashZone NextGen administrators can also disable or change the default availability for specific operations in MashZone NextGen Server configuration.

- **Data Altering Operations**: deletetable-name, inserttable-name and updatetable-name

  Operations that directly alter data are enabled by default for tables that are included in database mashables.

- **Finders**: findtable-nameAll, findtable-nameByPrimaryKey, findtable-nameWhere, findtable-nameByWhereClause, findtable-nameWherecolumn-nameEquals and selecttable-name.

  The finders `select`*`table-name`*`, findtable-name Where` and `findtable-name`*`ByWhereClause`* allow you to define arbitrary SQL queries. You can also define custom finders. See "Arbitrary SQL Queries for Database Mashables" on page 124 for more information.

If tables or stored procedures support overloaded operations, second and subsequent variations generate a mashable operation with the same name plus a unique index number. For example, `findEmployeeByPrimaryKey` and `findEmployeeByPrimaryKey0`.

Note: Sample requests and responses in this topic are based on a database table with these names:

- Database table name = EMPLOYEES

- Mashable table name = Employees
- Mashable name = EmployeeTable

delete*table-name*

Deletes one record from a database table based on a primary key.

Input Parameters	primaryKey
Return Value	none

Sample Request	`{ "version":"1.1",`
	`"sid":"EmployeeTable",`
	`"svcVersion":"0.1",`
	`"oid":"deleteEmployees",`
	`"header":{ "resultFormat":"json" },`
	`"params": [ {"primaryKey": 205} ]`
	`}`

find*table-name* All

Returns all records from this table. If this operation is called within a mashup that has previously set a maximum number of rows to return, the result set is limited to that maximum.

Input Parameters	None
Result Set	*table-name* _Array object with an array of all allowed records.

Sample Response	`{ "version":"1.1",`
	`"sid":"EmployeeTable",`
	`"appId":"",`
	`"oid":"findEmployeesAll",`
	`"svcVersion":"0.1",`
	`"header":{ "map":{ "serviceHeader":{ "map":{ } } } },`
	`"error":null,`
	`"errorCode":"",`
	`"invId":"",`
	`"response":"{ "Employees_Array": {`
	`"Employees":[`
	`{ "departmentId":90,`
	`"hireDate":{"time":5.509116e11}`
	`"email":"SKING",`
	`"employeeId":100,`
	`"jobId":"AD_PRES",`
	`"phoneNumber":"515.123.4567",`
	`"firstName":"Steven",`
	`"lastName":"King"`
	`},`
	`{ "departmentId":90,`
	`"hireDate":{"time":6.223644e11}`
	`"email":"NKOCHHAR",`
	`"employeeId":101,`
	`"jobId":"AD_VP",`

Input Parameters	None

	<pre>"phoneNumber":"515.123.4568",
	"firstName":"Neena",
	"lastName":"Kochhar"
	},
	...
	] }
	}"
	}</pre>

find*table-name* ByPrimaryKey

Returns one record or none that matches the specified primary key.

Input Parameters	primaryKey
Result Set	Object with matching record.

Sample Request	<pre>{ "version":"1.1",
	"sid":"EmployeeTable",
	"svcVersion":"0.1",
	"oid":"findEmployeesByPrimaryKey",
	"header":{ "resultFormat":"json" },
	"params": [{"primaryKey": 205}]
	}</pre>

Sample Response	<pre>{ "version":"1.1"
	"sid":"EmployeeTable",
	"appId":"",
	"oid":"findEmployeesByPrimaryKey",
	"svcVersion":"0.1",
	"header":{ "map":{ "serviceHeader":{ "map":{ } } } },
	"error":null,
	"errorCode":"",
	"invId":"",
	"response":{
	"departmentId":50,
	"hireDate":{"time":6.223644e11}
	"email":"MWEISS",
	"employeeId":120,
	"jobId":"ST_MAN",
	"phoneNumber":"650.123.1234",
	"firstName":"Matthew",
	"lastName":"Weiss" }
	}</pre>

find*table-name* ByWhereClause

Returns records that match the WHERE clause specified as a parameter. Dynamic parameters must be used in the WHERE clause with values specified in the second parameter as a comma-separated string.

This is an optional finder that is generated by default for tables and views. See "Arbitrary SQL Queries for Database Mashables" on page 124 for more information on the implications of using this finder.

Input Parameters	■ whereClause as a string with SQL code using ? for dynamic parameters. ■ params as a string with a list of values, separated by commas, to substitute for each dynamic parameter. Values are listed in the order they should be placed in the WHERE clause. **Note:** All substitution values must convert from string. Values that contain commas are not supported.
Result Set	*table-name* _Array object with all matching records.

Sample Request	`{ "version":"1.1",`
	`"sid":"EmployeeTable",`
	`"svcVersion":"0.1",`
	`"oid":"findEmployeesByWhereClause",`
	`"header":{ "resultFormat":"json" },`
	`"params": [ "FIRST_NAME LIKE ?", "L%" ]`
	`}`

find*table-name* Where

Returns records that match the WHERE clause specified as a parameter. No dynamic parameters can be used in the WHERE clause.

This operation uses raw SQL rather than a prepared statement which may be a security concern in some environments. It is disabled, by default. See "Arbitrary SQL Queries for Database Mashables" on page 124 for more information.

Input Parameters	whereClause as a string with SQL code.
Result Set	*table-name* _Array object with all matching records.

Sample Request	`{ "version":"1.1",`
	`"sid":"EmployeeTable",`
	`"svcVersion":"0.1",`
	`"oid":"findEmployeesWhere",`
	`"header":{ "resultFormat":"json" },`
	`"params": [ "FIRST_NAME LIKE 'L%'" ]`
	`}`

find*table-name* Where*column-name* Equals

One operation for each column that is configured for the tables or views in the mashable information source. Returns one or more records from a table where the column value equals the specified parameter.

Input Parameters	*column-name* specified in the type that matches this column
Result Set	*table-name* _Array object with all matching records.
Sample Request	`{ "version":"1.1", "sid":"EmployeeTable", "svcVersion":"0.1", "oid":"findEmployeesWhereEmployeeIdEquals", "header":{ "resultFormat":"json" }, "params": [ "205" ] }`

insert*table-name*

Inserts one record to a database table.

Note: This operation only updates the columns that are configured in the mashable information source for the table.

Input Parameters	dataTransferObject with properties for each mashable column for the record to insert.
Return Value	The primary key of the new record
Sample Request	`{ "version":"1.1", "sid":"EmployeeTable", "svcVersion":"0.1", "oid":"insertEmployees", "header":{ "resultFormat":"json" }, "params": [{ "departmentId":50, "hireDate":{"time":6.223644e11} "email":"NEWPERSON", "employeeId":230, "jobId":"ST_MAN", "phoneNumber":"510.123.1234", "firstName":"New", "lastName":"Person" }]`

select*table-name*

Performs arbitrary queries on this table using the SELECT clause and optional WHERE clause specified as parameters.

This operation uses raw SQL rather than a prepared statement which may be a security concern in some environments. It is disabled, by default. See "Arbitrary SQL Queries for Database Mashables" on page 124 for more information.

Input Parameters	■ selectClause as a string with raw SQL code to select all or specific columns from records in this table

	■ optional whereClause as a string with raw SQL code to specify which records to return
Result Set	Rows object with an array of objects containing the selected columns for all records that match the whereClause.

Sample Request	<pre>{ "version":"1.1",
	  "sid":"EmployeeTable",
	  "svcVersion":"0.1",
	  "oid":"selectEmployees",
	  "header":{ "resultFormat":"json" },
	  "params": ["EMPLOYEE_ID, FIRST_NAME, EMAIL",
	  "FIRST_NAME LIKE 'L%'"]
	}</pre>

Sample Response	<pre>{ "version":"1.1"
	  "sid":"EmployeeTable",
	  "appId":"",
	  "oid":"selectEmployees",
	  "svcVersion":"0.1",
	  "header":{ "map":{ "serviceHeader":{ "map":{ } } } },
	  "error":null,
	  "errorCode":"",
	  "invId":"",
	  "response": { "rows":[
	  { "FIRST_NAME":"Lex",
	  "EMPLOYEE_ID":"102",
	  "EMAIL":"LDEHAAN" },
	  { "FIRST_NAME":"Luis",
	  "EMPLOYEE_ID":"113",
	  "EMAIL":"LPOPP" },
	  ...
	  ]
	  },
	}</pre>

update*table-name*

Updates the record identified by primary key with the data object passed.

Note: This operation only updates the columns that are configured in the mashable for the table.

Input Parameters	■ primaryKey for the record to update ■ dataTransferObject with properties for each mashable column in the record to update.
Return Value	None

Sample Request	<pre>{ "version":"1.1",
	  "sid":"EmployeeTable",
	  "svcVersion":"0.1",
	  "oid":"updateEmployees",
	  "header":{ "resultFormat":"json" },
	  "params": [{"employeeId": 101},

```
      { "departmentId":90,
        "hireDate":{"time":6.223644e11}
        "email":"NKOCHHAR",
        "employeeId":101,
        "jobId":"AD_VP",
        "phoneNumber":"408.987.6543",
        "firstName":"Neena",
        "lastName":"Kochhar"
      } ]
}
```

Arbitrary SQL Queries for Database Mashables

You can configure custom database mashables to handle arbitrary queries using finders that are built into MashZone NextGen. You can also define custom finders to support specific, custom queries. This topic discusses the pros and cons of each approach.

Built-in Finders for Arbitrary Queries

Database mashables can include these built-in operations to handle arbitrary SQL queries for a single table or view:

- `find`*`table-name`*`ByWhereClause`:
  - Accepts a parameter with the SQL code for a WHERE clause which can contain dynamic parameters (as `?`). This can also include an ORDER BY clause at the end.

    For example: `"LAST_NAME LIKE ? AND DEPT_NO = ?"`
  - Accepts a second parameter with a string containing the values for all dynamic parameters in the WHERE clause. Parameters are separated by commas. They must be simple values that cannot contain commas.

    For example: `"L%,1020"`
  - Uses a prepared statement and thus has no risk of SQL injection attacks.
  - Returns all columns defined for the mashable.
- `find`*`table-name`*`Where`:
  - Accepts a parameter with the SQL code for a fully-specified WHERE clause. This can also include an ORDER BY clause at the end.
  - Should be used only by users familiar with the database schema as parameters use the database SQL names for columns rather than mashable names.
  - Executes raw SQL and thus is vulnerable to SQL injection attacks.
  - Returns all columns defined for the mashable.
- `select`*`table-name`*
  - Accepts a parameter with the SQL code for a SELECT statement for the specific table. You cannot specify other table names.

    For example: `"EMPLOYEE_ID, FIRST_NAME, EMAIL"`

- Accepts a parameter with the SQL code for a fully-specified WHERE clause.

  For example: `"FIRST_NAME LIKE 'L%'"`

- Should be used only by users familiar with the database schema as parameters use the database SQL names for columns rather than mashable names.
- Executes raw SQL and thus is vulnerable to SQL injection attacks.
- Can select specific columns from those defined for the mashable.
- Can sort results.

By default, `find`*`table-name`*`Where` and `select`*`table-name`* *are not enabled* when you register a database mashable. MashZone NextGen administrators can disable both these operations or enable them by default individually.

Custom Queries

You can also define custom finders for specific queries when you register database mashables. Custom finders:

- Accept a parameter with the SQL code for a WHERE clause. This can include one or more dynamic parameters which you define and map to mashable column names.

  For example: `"DEPT_NO = :deptNo AND HIRED_DATE < :hiredDate"`

- Dynamic parameters for the WHERE clause must be simple types with corresponding Java and JDBC types.
- Use prepared statements and thus have no risk of SQL injection attacks.
- Return all columns defined for the mashable.
- Can sort results by one or more columns that you specify.

Database Mashables Responses

Responses for database mashable operations may be a single return value, a result set or a combination of parameters, return value, and result sets. In many cases, MashZone NextGen determines the structure of the response when you register a database mashable.

The different structures and names that can occur in database responses include:

- **Well-Known Result Sets**: this response structure is used for all custom finders and all built-in finders, with the exception of the built-in `select`*`table-name`* finder. The complete structure of the result set is well defined when the mashable is registered.

  A result set in JSON format would look like this:

```
{ "sid": database-service-name
    "oid: finder-name
    ...
    "response": {"records": {
      "record": [
        { "mashable-column-name": columln-value,
          "mashable-column-name": columln-value,
```

```
          ...},
        { //objects for each row in result set}, ... },
        ...
      ]
    }
  }
}
```

- The result set is enclosed in an object named `records`.
- `records` contains a property named `record` that is an array of row objects.
- Each row object contains properties for each column in the result set.
- Table, view and column names are from the database mashable, *not* from the database.

For examples of responses with well-known results sets, see the `find`*table-name* `All` finder in the Default Operations Available for Database Mashables topic.

- **Dynamic Result Sets**: this may be the entire response for a finder, such as the built-in `select`*table-name* finder, or one result set returned along with parameters or a return value from a stored procedure. With dynamic result sets, MashZone NextGen does not know what the result set structure is when you register the database mashables.

  Result sets for stored procedures are commonly dynamic. For the `select`*table-name* finder, a parameter dynamically defines the columns for the result set.

  The pattern of a dynamic result set response is:

```
{ "sid": database-service-name
  "oid: finder-name
  ...
  "response": {"rows": [
    { "sql-column-name": columln-value,
      "sql-column-name": columln-value,
      ...}
        { //objects for each row in result set}, ... },
        ...
      ]
    }
  }
}
```

  The result set is enclosed in an object named `rows` which contains an array. The array contains objects for each row and those objects have properties for each column in the result set. Column names are from the database – not from the database mashable.

  See the sample response for `select`*table-name* finder in the Default Operations Available for Database Mashables topic for an actual example.

- **REF CURSORs**: are available only for Oracle databases. REF CURSORs may be the entire response for a packaged function or they can be one or more parameters returned from a stored procedure.

  REF CURSORs are treated exactly like dynamic result sets. The REF CURSOR is enclosed in an object named `rows` which contains an array. The array contains

objects for each row and those objects have properties for each column in the records for the REF CURSOR. Column names are from the database – not from the database mashable.

See "Example 1. Stored Procedure Examples" on page 127 for examples.

- *INOUT or OUT Parameters*: stored procedures may also return parameters with simple values if they are INOUT or OUT parameters. Currently for Oracle databases only, OUT parameters may also be REF CURSORs.

  Parameters are returned in the response in a property with the same name. See "Example 1. Stored Procedure Examples" on page 127 for examples.

- **Return Values**: from stored procedures are simple integers. They are returned in the response as a property named `returnValue`. See "Example 1. Stored Procedure Examples" on page 127 for an example.

- **Unknown Return Type**: this return type happens with stored procedures when JDBC metadata defines the return type as unknown. These are handled very much like dynamic result sets, but the structure of the response includes a `DynaBean` element. See "Example 1. Stored Procedure Examples" on page 127 for an example.

Stored Procedure Examples

The following example shows a dynamic response with a return value, two parameters with simple values and a result set:

```
{ "sid":"someDBSvc",
  "oid":"executeSomeProcedure",
  ...
  "response": {
    'returnValue':1,
    'OUT_PARAM':'abc',
    'INOUT_PARAM':'def',
    'rows':[
       { 'ID':15, 'PART_NO':'ABC123', 'QTY':0 },
       { 'ID':167, 'PART_NO':'ABC456', 'QTY':120 },
       { 'ID':201, 'PART_NO':'ABC789', 'QTY':34 }
    ]
  }
}
```

The next example shows a response from an Oracle package function with a REF CURSOR return type:

```
{ "sid":"someDBSvc",
  "oid":"executePkgSomeFunction",
  ...
  "response": {
    'rows':[
       { 'ID':15, 'PART_NO':'ABC123', 'QTY':0 },
       { 'ID':167, 'PART_NO':'ABC456', 'QTY':120 },
       { 'ID':201, 'PART_NO':'ABC789', 'QTY':34 }
    ]
  }
}
```

The next example shows REF CURSORS returned from an Oracle procedure in OUT parameters:

```
{ "sid":"someDBSvc",
  "oid":"executeAnotherProcedure",
  ...
  "response": {
    'OUT_PARAM1':{
        'rows': [{
            'ITEM1':'value 1',
            'ITEM2':'value 2',
            'ITEM3':'value 3'
        }]
    },
    'OUT_PARAM2':{
        'rows': [{
            'ITEM7':'value 7',
            'ITEM8':'value 8',
            'ITEM9':'value 9'
        }]
    }
  }
}
```

The final example shows a response for a stored procedure with an unknown return type. It actually returns both a return value and a result set:

```
{ "sid":"someDBSvc",
  "oid":"executeSomeProcedure",
  ...
  "response": {
    'DynaBean': {
      'returnValue': 'value 0',
      'rows':[ {
        'ITEM1':'value 1',
        'ITEM2':'value 2',
        'ITEM3':'value 3'
      } ]
    }
  }
}
```

MashZone Feeds

MashZone feeds are created in Software AGMashZone to retrieve data from a variety of sources and, like mashups, convert or transform that data. MashZone feeds can get data from many of the same types of sources as MashZone NextGen mashables such as CSV files, Excel spreadsheets and relational databases.

Note: For more information on the types of data sources you can use in MashZone feeds, see *Data Sources* in the MashZone documentation. MashZone feeds use the Integrated MashZone Server for MashZone NextGen. MashZone NextGen administrators must first Manage the Integrated MashZone NextGen Server Connection before you can work with MashZone feeds.

Once this initial configuration is complete, MashZone NextGen administrators and administrators for your existing MashZone Servers can export existing feeds from MashZone and import them into MashZone NextGen.

Users can edit existing feeds, adding views and adding them to apps or workspace apps.

If MashZone NextGen administrators have enabled this, users may also be able to create new MashZone feeds in the MashZone Feed Editor in MashZone NextGen. See "Create or Edit MashZone Feeds" on page 159 for instructions.

REST Web Services

REST, or *representational state web services*, are web services available via HTTP or HTTPS that use URLs or the body of POST requests to invoke service operations and pass input parameters. REST web services may use either the GET or POST operations in HTTP.

Note: You can also register syndicated feeds (RSS or Atom) as REST web services.

MashZone NextGen uses the following operation names for REST web services based on the request method of the service:

- `getData`: for GET requests.
- `postData`: for POST requests.

Like several other types of mashables, MashZone NextGen supports secure communication with REST web services with these security profiles:

- Basic HTTP Authentication
- NTLM Authentication
- SSL

MashZone NextGen administrators can also add custom security mechanisms to handle specific information source security requirements. See "Configure Secure Connections for Mashables" on page 164 for more information.

MashZone NextGen administrators can also control the HTTP response headers for REST web services or timeouts for connections with REST web services.

You can also configure the HTTP request header for REST web services. See "Configure HTTP Request Headers" on page 174 for more information.

For information on registering, see "Register REST Web Services" on page 138.

GET-Based REST Web Services

Most REST web services accept requests that use the GET operation in HTTP to invoke the service. Parameters are passed within the URL to the service following the question mark (?) delimiter. The URL for a REST web service using GET might look like this:

```
http://www.someOrg.com/myRestService?date=20080301&zip=94102
```

When you register REST web services in MashZone NextGen, you provide the URL to the web service. For GET-based REST services, the URL should include any parameters that you need to use for any request. You can also include default values for these parameters or simply omit the value.

Note: MashZone NextGen uses the default parameter values that you supply when you register a REST web service to validate the URL during registering. MashZone NextGen*also* uses the default parameter values when the request to run the service contains *no parameters* at all.

POST-Based REST Web Services

Some REST web services, however, use the POST operation in HTTP to invoke the service. Parameters are passed in the body of the POST request.

MashZone NextGen supports any valid format for the body of a POST request and the appropriate MIME types. Common examples include XML, JSON or plain text.

The URL for a REST web service using POST might look like this:

```
http://www.someOrg.com/myRestService
```

Snapshot Mashables

Snapshots save the specific results when a mashable or mashup is run. To use a snapshot in a basic app, share a snapshot with other users or easily use a snapshot in a mashup in Wires, you must Register a Snapshot as a Mashable.

Note: MashZone NextGen developers can use snapshots directly in mashups using EMML and the MashZone NextGen Snapshot API. Developers can also create custom apps that use snapshots directly through this API.

You can also Take, View or Delete Snapshots or Schedule Snapshots from any mashable or mashup artifact page.

Spreadsheet Mashables

You can use Excel spreadsheets as a mashable information source in MashZone NextGen by registering cell ranges as:

- A snapshot of data for Excel spreadsheets on your local computer or any accessible network drive. With local spreadsheets, you must reregister to update the spreadsheet mashable in MashZone NextGen if you update spreadsheet data and want that information to be accessible in MashZone NextGen.
- A dynamic source of data for Excel spreadsheets that are accessible remotely via HTTP. Remote access allows MashZone NextGen to retrieve current data whenever the mashable information source is used.For example, spreadsheets hosted in SharePoint can be registered remotely.

  Note: Remote spreadsheet mashables are handled as REST mashables in MashZone NextGen. With remote spreadsheet mashables, MashZone NextGen administrators can also control HTTP response headers and timeouts for connections.

See "Register an Excel Worksheet File from MashZone NextGen Hub" on page 143 for more information about registering spreadsheets. See "Valid Labels and Values for Spreadsheet/CSV Mashables" on page 131 also.

Valid Labels and Values for Spreadsheet/CSV Mashables

Some labels and data that are valid in CSV files or Excel spreadsheets are not valid or not accessible in MashZone NextGen. In the following cases, MashZone NextGen may alter labels or transform or ignore the information:

- **Formulas** in spreadsheets are valid, but the formula itself is not exported to the mashable in MashZone NextGen. The result of the calculation (the data) is exported instead.
- **Column labels**:
  - Are recommended, but optional, for each column that contains data.

    If column labels are missing, MashZone NextGen automatically generates labels in the form `Column1, Column2` and so on.
  - Are limited to letters, numbers and underscores in the ISO-8859-1 character set, also known as Latin 1.

    Spaces and most symbols or punctuation characters are either converted to an underscore or ignored.
  - Must start with a letter (an alphabetical character) although they can contain numbers after that. Column labels that do not start with a letter are altered or overridden.
  - If a column label starts with a number or is a formatted date, MashZone NextGen adds a prefix of `Column_` to the label.
  - Can only be provided by *one* row. For CSV files, this must be the first row.

    For spreadsheets you can identify which row supplies the column labels when you register the mashable. If your spreadsheet has column names in two or more rows, you must choose a single row, allow MashZone NextGen to generate column labels or simplify your worksheet.
  - Cannot span two or more cells in a spreadsheet. MashZone NextGen treats the second and subsequent columns as missing labels. You can allow MashZone NextGen to generate column names for the second and subsequent columns or you can update your worksheet.
- **Charts, images or other objects** in spreadsheets are ignored and not saved in mashable data.

Syndicated Feeds (RSS/Atom)

RSS, or Really Simple Syndication, refers to *syndicated web feeds* that use either the RSS or Atom standard formats to send messages to subscribers. In MashZone NextGen, RSS is

the mashable type for these web feeds regardless of the actual format that the feed uses. For information on registering RSS or Atom feeds, see "Register Syndicated Feeds (RSS/Atom)" on page 137.

MashZone NextGen can be configured to *normalize* the responses for web feeds to a single format, making it easier to use these responses in mashups. See "Normalization and MashZone NextGen Support for RSS/Atom Formats" on page 132 for more information on normalization and the syndication formats that MashZone NextGen supports.

RSS mashables may use HTTP or HTTPs. Like some other mashable information sources, MashZone NextGen supports secure communication with RSS mashables using these security profiles:

- Basic HTTP Authentication
- NTLM Authentication
- SSL

MashZone NextGen administrators can also add custom security mechanisms to handle specific information source security requirements. See "Configure Secure Connections for Mashables" on page 164 for more information.

MashZone NextGen administrators can also control the HTTP response headers for syndicated web feeds or timeouts for connections.

Normalization and MashZone NextGen Support for RSS/Atom Formats

MashZone NextGen*no longer* normalizes syndicated feed results to a single standard format and version, as it did in earlier releases. MashZone NextGen administrators can change configuration to enable normalization if needed. Or MashZone NextGen developers can request normalization for a specific invocation of a web feed using MashZone NextGen Headers/Parameters.

Although normalization can make using information from syndicated feeds easier, there are limitations:

- Syndicated feeds in Atom formats *cannot* be normalized to the RSS format and vice versa.

  If you need normalization and also need syndicated feeds in both RSS and Atom formats, you can normalize to the RSS format and publish feeds that use the Atom format as REST web services instead. Or, normalize to Atom and publish RSS feeds as REST web services.

- Some syndicated feeds extend the standard format to include additional information, such as GEORSS which includes geographical location data. In these cases, normalization may remove extension information that you want.

  With normalization enabled, you can publish feeds that use GEORSS or other extended syndication formats as REST web services. Or you can disable normalization (set the normalization type to `native`).

- Syndicated responses in languages other than English may be scrambled by normalization. The character encoding is affected in some cases, making the responses unreadable. In this case, the solution is to leave normalization disabled.

WSDL Web Services

MashZone NextGen WSDL mashables are SOAP web services using SOAP version 1.1 or 1.2 in document style or RPC style. For more information, see:

- "WSDLs for SOAP Web Services" on page 133
- "Security Policies and Configuration" on page 133
- "WSDL Web Service Responses" on page 134

See also "Register WSDL Web Services" on page 141.

WSDLs for SOAP Web Services

SOAP web services use WSDLs (web service definition language files) to define the service contract. When you register a WSDL web service, MashZone NextGen retrieves the WSDL, including any schemas that are included in the WSDL, to help define meta-data for the WSDL mashable.

You must provide a URL to the primary WSDL when you register the mashable. MashZone NextGen supports WSDLs that include schemas *only when* the locations for included fragments are relative to the location of the primary WSDL or that have absolute addresses. For some SOAP web services you may also need to define security configuration for the web service to allow MashZone NextGen to access the service.

If your environment does not have Internet access or you experience other WSDL access problems when registering a WSDL mashable, you may need to download the primary WSDL and any included schemas to the MashZone NextGen configuration folder or the file system of the MashZone NextGen Server host. You may also need to edit the primary WSDL to reflect the new locations of included schemas.

Some WSDLs use array datatypes from the SOAP Encoding namespace (`http://schemas.xmlsoap.org/soap/encoding`) but do not actually import this schema. This can cause unknown type or type resolution errors when you register the WSDL web service. You can set configuration for the MashZone NextGen Server to work around this issue.

Security Policies and Configuration

MashZone NextGen supports Web Services Security (WSS) to define security constraints and policies for WSDL web services. See "Web Services Security (WSS) for WSDL Services" on page 134 for more information.

For SOAP web services that do not use WSS, you can set up secure communications between MashZone NextGen and the web service using these security profiles:

- Basic HTTP Authentication, for service invocation only
- NTLM Authentication, for both web service invocation and retrieval of the WSDL during service registering
- SSL, for both web service invocation and retrieval of the WSDL during service registering

MashZone NextGen administrators can also add custom security mechanisms to handle specific information source security requirements. See "Configure Secure Connections for Mashables" on page 164 for more information.

MashZone NextGen administrators can also control the HTTP response headers for WSDL web services or timeouts for connections with WSDL web services.

WSDL Web Service Responses

Responses for WSDL services are typically XML, defined in the service's WSDL. Some WSDL services return HTML. MashZone NextGen currently only supports HTML results for WSDL services if the HTML payload is wrapped in a CDATA section. For WSDL services that do not use a CDATA section to wrap HTML, you can use a mashup to invoke the service and handle the response properly. See "Wrapping WSDL Web Services That Return HTML Results" on page 628 for more information.

Web Services Security (WSS) for WSDL Services

Web Services Security (WSS) encompasses a set of extensible standards to allow SOAP web services to define security constraints and requirements so that client applications can determine and understand them programmatically. It includes many of the *WS-** standards, co-authored by several well known software vendors, plus standards from W3C and OASIS.

To define its WSS security requirements for MashZone NextGen, a web service must provide this information:

- **Policy assertions**: that define a specific security constraint, such as authentication and identification requirements, message integrity or confidentiality requirements, wire protocols, message exchange patterns, digital signatures, cryptographic requirements and so on.
- **Policies**: that define how *policy assertions* are combined and applied for a specific use. See "Built-in WSS Support for Policies and Policy Attachments" on page 135 for an example of a simple policy declaration.
- **Policy attachments**: that define which policies are used to secure specific components in the web service. Policies can be attached to the service as a whole, to endpoints, operations, messages or elements within a message.

Web service clients must include SOAP headers in requests to provide the required encryption, tokens, certificates, digests or other artifacts needed to comply with the

policies for the web service. MashZone NextGen can automatically generate the WSS SOAP headers needed for WSDL web services that explicitly define security policies.

To support the very broad set of features and the extensible nature of WSS, MashZone NextGen uses an extensible architecture with built-in support for some simple, common security policies. See "Built-in WSS Support for Policies and Policy Attachments" on page 135, "Built-in WSS Support for Policy Assertions and Tokens" on page 136 and "Built-in WSS Support for Certificates and Encryption" on page 136 for specific information on the WSS features that MashZone NextGen supports 'out of the box.'

MashZone NextGen can also be extended to support additional policy profiles or security standards. Please contact your Software AG representative for more information on WSS extensions.

MashZone NextGen also supports authentication for WSDL web services that do not use WSS policies using HTTP basic authentication, Windows NT Domains or SSL with digital certificates. See "WSDL Web Services" on page 133 for links to more information.

Built-in WSS Support for Policies and Policy Attachments

MashZone NextGen supports the following policy framework and attachment specifications:

- Web Services Policy Framework 1.2
- Web Services Policy Attachments 1.2

Both specifications have been submitted to the W3C for adoption as standards, a work in progress. For schema information on the 1.2 standards, see http://schemas.xmlsoap.org/ws/2004/09/policy/. For information on W3C policy drafts based on these specifications, see http://www.w3.org/2002/ws/policy/.

Out of the box, MashZone NextGen can generate one SOAP header for policies and attachments in these cases:

- Policies must be defined in the primary WSDL for a WSDL web service.
- Policies *cannot* be defined in imported fragments or referenced by absolute URLs.
- A policy *must* be defined at the global level in the WSDL for the entire web service, as a direct child of the <definitions> element. Policies for other service components are not supported 'out of the box.'
- Only one policy can apply to a WSDL web service component. Policy merges are not supported.

A WSDL for a web service might look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="SomeServiceDefinition"
    targetNamespace="http://someCompany.com/webServices/SomeService"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:s0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
       wss-wssecurity-utility-1.0.xsd"
    xmlns:s1="http://someCompany.com/webServices/SomeService"
    xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:UsingPolicy n1:Required="true"
      xmlns:n1="http://schemas.xmlsoap.org/wsdl/"/>
  <wsp:Policy s0:Id="authPolicy">
    <wssp:Identity xmlns:wssp="http://www.softwareag.com/presto10/security/policy">
      <wssp:SupportedTokens>
        <wssp:SecurityToken
            TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
            wss-x509-token-profile-1.0#X509v3"/>
      </wssp:SupportedTokens>
    </wssp:Identity>
  </wsp:Policy>
  <types>
...
```

Built-in WSS Support for Policy Assertions and Tokens

MashZone NextGen support for policy assertions and tokens is based on OASIS standards for WSS. See http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss for more information.

- OASIS Username Token Profile 1.0
- OASIS x509 Certificate Token Profile 1.0
- BEA Certificate Token Profile, which is based on the OASIS certificate profile. For more information, see http://e-docs.bea.com/wls/docs100/webserv_sec/index.html and http://e-docs.bea.com/wls/docs100/webserv_ref/sec_assert.html.
- Microsoft Web Services Enhancements (WSE) 3.0 UsernameOverTransport profile

  For more information on WSE 3.0, see http://msdn.microsoft.com/en-us/library/aa139619.aspx.
- MashZone NextGen Username Token Profile 1.0, an implementation of the OASIS Username Token Profile 1.0

Built-in WSS Support for Certificates and Encryption

MashZone NextGen also supports:

- Self-signed digital certificates
- Digital certificates from trust authorities
- XML Encryption

Connect Information Sources as Mashables

The steps necessary to make information sources *mashable* and register them in MashZone NextGen depends on the type of information source:

- For common databases, documents or systems that use standard connections, you:
  - Register Syndicated Feeds (RSS/Atom)
  - Register REST Web Services

- Register WSDL Web Services
- Register Basic Database Mashables. For a wider range of features, MashZone NextGen administrators can Register Custom Database Mashables
- Register Spreadsheets as Mashables
- Register CSV Mashables
- Register XML Mashables
- Register SharePoint Lists

  Note: This type of mashable information source is only supported if your MashZone NextGen Administrator has installed the MashZone NextGen Add-On for SharePoint and has configured SharePoint connections for the MashZone NextGen Server.

- Register a Snapshot as a Mashable

- For events, you Subscribe to Events from the Event Bus or Apama.
- To work with MashZone data feeds, you Create or Edit MashZone Feeds.

You may also handle Mashable Authentication with Security Profiles.

Register Syndicated Feeds (RSS/Atom)

For syndicated feeds that use SSL endpoint security, a MashZone NextGen administrator must configure MashZone NextGen for SSL and digital certificates before you register the feed.

If normalization is enabled forMashZone NextGen to one of the RSS or Atom web feed standards, you can only register syndicated feeds that use a version of that standard. Register feeds in the alternate format as REST web services.

Note: In earlier releases, feed normalization was enabled by default. Normalization is disabled, butMashZone NextGen administrators can turn this on if desired.

To register synicated feeds

1. Select **Connect > Feed** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for this web feed.MashZone NextGenuses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Enter the full **Feed URL** to this feed.
4. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.

5. If this mashable information source requires authentication information when you run it, choose the **Security Profile** to use for authentication and complete the fields for this profile. See "Mashable Authentication with Security Profiles" on page 161 for details.
6. Click **Register**.

   MashZone NextGen validates the URL, registers the mashable and turns this mashable information source on (by default). Then, either:

   - Grant permissions to run this mashable to other users and groups:
     - Choose the mashable-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
     - Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
     - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
     - Save the permissions once the list is complete.
   - Or skip permissions for now and click **Finish**.

     To allow other users to work with this mashable, you must grant run permissions at a later time.

> **Note:** Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

Register REST Web Services

1. For REST web services that require SSL authentication, you or a MashZone NextGen administrator must configure MashZone NextGen for SSL and digital certificates before you register the service.
2. Select **Connect > REST Web Service** in the MashZone NextGen Hub main menu.
3. Enter a unique **Name** for the web service.MashZone NextGenuses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
4. Enter the full URL for this mashable and, if needed, set document properties for this web service. See "REST Web Service URLs with Parameters" on page 139, "REST Web Services with Input Documents" on page 140 and "Specify HTTP Request Headers" on page 140 for details.

5. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
6. If this mashable information source requires authentication information when you run it, choose the **Security Profile** to use for authentication and complete the fields for this profile. See "Mashable Authentication with Security Profiles" on page 161 for details.
7. Click **Register**.

   MashZone NextGen validates the URL, registers the mashable and turns this mashable information source on (by default). Then, either:

   - Grant permissions to run this mashable to other users and groups:
     - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
     - Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
     - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
     - Save the permissions once the list is complete.
   - Or skip permissions for now and click **Finish**.

     To allow other users to work with this mashable, you must grant run permissions at a later time.

> **Note:** Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

REST Web Service URLs with Parameters

If the URL for the web service contains all the parameters and information that the web service needs, MashZone NextGen validates any parameters and values you supply in this URL. They are also used to generate metadata for the mashable. When users or apps run the mashable, MashZone NextGen sends a request to the web service using the HTTP GET operation.

> **Tip:** It is a good practice to list all possible parameters even though individual requests can omit parameters that are not needed for that request.

See also "GET-Based REST Web Services" on page 129 for more information on parameters and default values.

REST Web Services with Input Documents

If the REST web service needs more complex input that cannot be passed in the URL, you must set document properties for this mashable. When users or apps run the mashable, MashZone NextGen sends a request to the web service using the HTTP POST operation.

Set the **Requires Input Document** option. Then choose the correct MIME type for this web service as the **Document Type**.

- `text/xml` = for input in an XML format.
- `text/json` = for input in a JSON (JavaScript Object Notation) format.
- `text/plain` = for input that is just text.
- `url-form-encoded` = for input that comes from an HTML <form> or is in the form of name/value pairs.
- Or enter the appropriate MIME type

Enter or paste a **Sample Document** in the format required for the input document that this web service expects. Provide default values for fields as needed.

MashZone NextGen validates the sample input document that you supply and the default values. They are also used to generate metadata for the mashable.

Specify HTTP Request Headers

You can configure HTTP Request Headers optionally. HTTP Request Headers define the operating parameters of an HTTP transaction. The **Specify HTTP Request Headers** option allows you to add one or more HTTP Request Headers as Name:Value pairs. The **Name** field provides a list of http headers as a drop-down menu, but also allows you to type in a valid parameter. Depending on the selected parameter, the **Value** field provides a list of appropriate values as a drop-down menu or you can type in a proper value.

Click the + button and fill in the required `Name` and `Value`.

Tip: See List of HTTP header fields on Wikipedia.org

After the Registration of a REST Web Service you can still configure the HTTP Request Headers in the MashZone NextGen Hub.

See "Configure HTTP Request Headers" on page 174

Register WSDL Web Services

For WSDL web services that require SSL authentication, you or a MashZone NextGen administrator must configure MashZone NextGen for SSL and digital certificates before you register the service.

1. Select **Connect > WSDL Web Service** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for the web service.MashZone NextGenuses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Enter the full **Web Service Descriptor (WSDL) URL** to this web service.

   This URL points to the WSDL for this web service. If the WSDL includes external schemas, these must also be accessible by relative or absolute URL. See "WSDLs for SOAP Web Services" on page 133 for more information on MashZone NextGen support for WSDL locations and access.

   Note: DOCTYPE processing for WSDL is disabled by default to prevent External Entity Injection attacks. Registering a WSDL containing a DOCTYPE declaration will result in an error message. To enable DOCTYPE processing for WSDL, an administrator must set the `enable.wsdl.doctype` property in the presto.config file to `true`:

   ```
   #
   # parameter to enable DOCTYPE processing within WSDL documents. For security reasons
   # this should be set to false. Setting this to 'true' allows presto to process WSDL
   # documents containing DOCTYPE sections, but also then opens the possibility of
   # External XML Entity Injection attacks.
   #
   enable.wsdl.doctype = false
   ```

4. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
5. If this mashable information source requires authentication information when you run it, choose the **Security Profile** to use for authentication and complete the fields for this profile. See "Mashable Authentication with Security Profiles" on page 161 for details.
6. Click **Register**.

MashZone NextGen validates the URL, registers the mashable and turns this mashable information source on (by default). This also loads the WSDL in MashZone NextGen.

Then, either:

- Grant permissions to run this mashable to other users and groups:
  - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
  - Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
  - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
  - Save the permissions once the list is complete.
- Or skip permissions for now and click **Finish**.

  To allow other users to work with this mashable, you must grant run permissions at a later time.

Note: Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

Register Spreadsheets as Mashables

You can use Excel spreadsheets as mashable information sources in MashZone NextGen.

Important: How MashZone NextGen accesses spreadsheets determines whether the data from the spreadsheet is a simple snapshot or is dynamic.

If the spreadsheet is located on your computer or on a network drive, MashZone NextGen takes a snapshot of the data when you register the spreadsheet. Updates to your spreadsheet are *not* automatically visible in MashZone NextGen. You must re-register the spreadsheet to update the data in MashZone NextGen in this case.

If the spreadsheet is hosted in a content management system, such as SharePoint, or is simply accessible with a URL using HTTP, MashZone NextGen links to the spreadsheet rather than copying data. Each time users use the spreadsheet mashable in MashZone NextGen, MashZone NextGen retrieves the current data from the spreadsheet dynamically.

For more information about spreadsheet mashables, see "Spreadsheet Mashables" on page 130.

You have two ways that you can register worksheets in MashZone NextGen:

- From the **Connect** menu in MashZone NextGen Hub. See "Register an Excel Worksheet File from MashZone NextGen Hub" on page 143 and "Register an Excel Worksheet With a URL from MashZone NextGen Hub" on page 144 for instructions.
- With the MashZone NextGen Add-On for Excel.

Register an Excel Worksheet File from MashZone NextGen Hub

Use this procedure for Excel spreadsheets that are located on your computer or a network drive.

1. Select **Connect > Excel File** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for the mashable for this spreadsheet.. MashZone NextGen uses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Set **Upload File** as the location of the spreadsheet.
4. Click **Browse** and find the Excel spreadsheet you want to use.
5. Enter the name or tab number for the **Worksheet** that contains the data you want to use in MashZone NextGen.

   For example:

   - `3` identifies the worksheet in the third tab of this spreadsheet.
   - `RawData` identifies the worksheet with the name "RawData".
6. Enter the **Cell Range** for the data that you want to use in this mashable in the form: *top-left-cell:bottom-right-cell* .

   For example, `A1:G88`.
7. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
8. Click **Register**.

   MashZone NextGen uploads a snapshot of the spreadsheet data, registers the mashable and turns this mashable information source on (by default). Then, either:

   - Grant permissions to run this mashable to other users and groups:
     - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.

- "Find Users and Groups for Permissions" on page 104 or "Choose MashZone NextGen Built-In Groups for Permissions" on page 105.
- Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
- Save the permissions once the list is complete.

- Or skip permissions for now and click **Finish**.

  To allow other users to work with this mashable, you must grant run permissions at a later time.

> **Note:** Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

Register an Excel Worksheet With a URL from MashZone NextGen Hub

Use this procedure for Excel spreadsheets that are accessible with a URL using HTTP, such as through a content management system.

1. Select **Connect > Excel File** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for the mashable for this spreadsheet.. MashZone NextGen uses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Set **Link to URL** as the location of the spreadsheet.
4. Enter the full **URL** to this Excel spreadsheet.
5. If this spreadsheet requires authentication information when you run it, choose the **Security Profile** to use for authentication and complete the fields for this profile. See "Mashable Authentication with Security Profiles" on page 161 for details.
6. Enter the name or tab number for the **Worksheet** that contains the data you want to use in MashZone NextGen.

   For example:

   - `3` identifies the worksheet in the third tab of this spreadsheet.
   - `RawData` identifies the worksheet with the name "RawData".
7. Enter the **Cell Range** for the data that you want to use in this mashable in the form: *top-left-cell:bottom-right-cell* .

   For example, `A1:G88`.
8. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
9. Click **Register**.

MashZone NextGen validates the URL, registers the mashable and turns this mashable information source on (by default). Then, either:

- Grant permissions to run this mashable to other users and groups:
  - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
  - "Find Users and Groups for Permissions" on page 104 or "Choose MashZone NextGen Built-In Groups for Permissions" on page 105.
  - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
  - Save the permissions once the list is complete.
- Or skip permissions for now and click **Finish**.

  To allow other users to work with this mashable, you must grant run permissions at a later time.

Register XML Mashables

You can register any well-formed XML file as a mashable information source. This creates a snapshot of the data in MashZone NextGen. If you update the data in the XML file, you must reregister it to MashZone NextGen to make those updates accessible in MashZone NextGen.

1. Select **Connect > XML** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for this mashable information source. MashZone NextGen uses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Enter the location of the XML file or click **Browse** to find the file in your computer or on a network drive.
4. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
5. If this XML file is accessed via HTTP and requires authentication information when you run it, choose the **Security Profile** to use for authentication and complete the fields for this profile. See "Mashable Authentication with Security Profiles" on page 161 for details.
6. Click **Register**.

   MashZone NextGen retrieves a snapshot of the XML file, registers the data as the mashable and turns this mashable information source on (by default). Then, either:

   - Grant permissions to run this mashable to other users and groups:

- Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
- Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
- Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
- Save the permissions once the list is complete.

- Or skip permissions for now and click **Finish**.

  To allow other users to work with this mashable, you must grant run permissions at a later time.

> **Note:** Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

Register CSV Mashables

You can register any file that uses the commas-separated-values (CSV) format as a mashable information source. Access to the data in the file depends on the file location.

1. Select **Connect > CSV** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for this mashable information source. MashZone NextGen uses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Choose the **Location of the CSV file**:
   - For files that are on your computer or a network drive, choose **Local**. Then enter the location for the file or click **Browse** to find the file.

     With local files, MashZone NextGen takes a snapshot of the data. To make updates to the data available, you must reregister local CSV files.
   - For files that are available via HTTP, via content management systems such as SharePoint, choose **Remote** and enter the **URL** to this file.

     For remote files, MashZone NextGen accesses the file dynamically each time the mashable is run. This ensures that updates to the data are always available.
4. If the CSV file uses a delimiter other than a comma, change the **Delimiter** to:
   - `Comma` (the default)

- `Tab`
- `Other` for any other character. Then enter the delimiter.

5. If the first line of the CSV file does *not* contain the names of the fields in each record, clear the **Use the First Row as Column Header** option.

   MashZone NextGen will generate field names such as Column1.

6. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.

7. If this CSV file is remote and requires authentication information when you run it, choose the **Security Profile** to use for authentication and complete the fields for this profile. See "Mashable Authentication with Security Profiles" on page 161 for details.

8. Click **Register**.

   MashZone NextGen retrieves a snapshot of the CSV file, registers the data as the mashable and turns this mashable information source on (by default). Then, either:

   - Grant permissions to run this mashable to other users and groups:
     - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
     - Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
     - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
     - Save the permissions once the list is complete.
   - Or skip permissions for now and click **Finish**.

     To allow other users to work with this mashable, you must grant run permissions at a later time.

> **Note:** Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

Register Basic Database Mashables

a MashZone NextGen administrator must configure the data source for this database. MashZone NextGen administrators should

You can register database mashables in MashZone NextGen for one table, view or stored procedure. The operations that MashZone NextGen generates for these mashable information sources depends on configuration defined by your MashZone NextGen administrator.

Typically, a database mashable includes queries or *finders* to find records based on any column in the table or view. They may also include operations to insert, update or delete records. Mashables for stored procedures have a single operation that runs that stored procedure. See "Default Operations Available for Database Mashables" on page 118 for more information.

Important: MashZone NextGen administrators have additional features they can use to register database mashables that work with combinations of database resources within a given schema or catalog in one data source. They can also customize the database mashable to fine tune the operations that are available or add custom queries for more complex requirements. Administrators should see "Register Custom Database Mashables" on page 149 for more information.

1. Select **Connect > Database** in the MashZone NextGen Hub main menu.
2. Enter a unique **Name** for this mashable information source. MashZone NextGen uses this name to assign a unique ID. For more information, see "What are valid artifact names and IDs?" on page 69.
3. Select the **Datasource** for this mashable.
4. Depending on the type of database, select a **Schema**, a **Catalog** or both.

   For most types of databases, you select a schema. If information for both fields is available, select both to narrow the list of database assets.

   Note: For MySQL databases, choose the Catalog entry that identifies the name of the database.

   After a few seconds, information for the tables, views and stored procedures in this schema or catalog for this data source are displayed.
5. Choose `Table`, `View` or `Procedure` and then select the specific database table, view or stored procedure that you want to use.

   By default, all columns for database tables are included in the database mashable. You can exclude certain columns, if needed:

   a. In the list of columns that opens, select the columns that you want to exclude from the database mashable.

   b. Click **Remove**.

   Note: You *cannot* delete the primary key column.

6. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
7. Click **Register**.

   MashZone NextGen registers the database mashable and turns this mashable information source on (by default). Then, either:

   - Grant permissions to run this mashable to other users and groups:
     - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
     - Find Users and Groups for Permissions or Choose MashZone NextGen Built-In Groups for Permissions.
     - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
     - Save the permissions once the list is complete.
   - Or skip permissions for now and click **Finish**.

     To allow other users to work with this mashable, you must grant run permissions at a later time.

> **Note:** Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

Register Custom Database Mashables

You or a MashZone NextGen administrator must configure the data source for this database.

Users can register basic mashable information sources for databases based on one table, view or stored procedure. MashZone NextGen administrators can register more complex, custom mashable information sources for databases based on any number of tables, views, stored procedures or combinations of these resources within a given schema or catalog in one data source.

Registering a mashable information source for a database generates a standard set of operations, based on database generator configuration. See "Default Operations Available for Database Mashables" on page 118 for more information.

You can fine tune the standard operations generated for a database mashable that you are registering. You can also add custom queries or *finders* for more complex database requirements.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand **Mashable Database Services** and click **Register Database Services**.
3. Click **Register new Database Service**.
4. Enter a unique **Name** for this mashable information source.

   MashZone NextGen uses the mashable name to assign a unique identifer to mashables. Mashable names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

   It is a good practice to name mashables based on their purpose, such as `AccountsPayable`, rather than on deployment or technical information. This will make it easier for other users to recognize and work with the mashable.
5. Enter an optional **Description** for the mashable information source.
6. Select the **Datasource** for this mashable.
7. Depending on the type of database, select a **Schema**, a **Catalog** or both.

   For most types of databases, you select a schema. If information for both fields is available, select both to narrow the list of database assets.

   > **Note:** For MySQL databases, choose the Catalog entry that identifies the name of the database.

   After a few seconds, information for the tables, views and stored procedures in this schema or catalog for this data source are displayed.
8. Check the tables, views or stored procedures that you want to expose in this mashable information source.
   - Click in the check box next to Tables, Views or Procedures to select all the listed artifacts.
   - Or, click individual tables, views or stored procedures.

   If you are not sure of exactly which tables, views or stored procedures you want, you can include them now, review their detail and then remove them from the mashable before you finish registering it.
9. Click **Proceed**.
10. Enable or disable operations with security implications; review table, view or stored procedure details; or update table or view details as needed to define the operations to include in this mashable:
    - Enabling/Disabling Data Alteration Operations and Other Security Considerations

- Viewing Details for Tables, Views or Stored Procedures
- Removing Tables, Views or Stored Procedures from the Mashable
- Removing Columns for Table or View Operations
- Changing Column Names or Datatypes for Tables or Views
- Removing Table or View Finders
- Changing Finders for Tables or Views
- Adding Finders for Tables or Views

11. Once the mashable operation details are complete, click **Proceed**.

MashZone NextGen registers the mashable and turns this mashable information source on (by default).

Enabling/Disabling Data Alteration Operations and Other Security Considerations

The mashable operations that have security implications include those that can alter data and those finders that use raw SQL rather than prepared statements. You manage these operations for the database mashable as a whole, rather than for individual tables or views.

By default MashZone NextGen generates mashable operations to insert, update or delete records for each table. You can disable any of these operations for all tables in the mashable.

The following finders are not generated by default. You can choose to enable these operations:

- **Dynamic Finder** = the `find`*table-name* `Where` operation. This executes a WHERE clause defined as a parameter.
- **Dynamic Select** = the `select`*table-name* operation. This executes a SELECT statement for the given table and optional WHERE clause defined as parameters.

These finders are very flexible, allowing you to perform arbitrary SQL commands. However, they are vulnerable to SQL injection attacks and thus a potential security risk. See "Arbitrary SQL Queries for Database Mashables" on page 124 for more information.

Administrators can also completely disable the use of these operations for new database mashables using MashZone NextGen Server configuration.

1. Select the **Database Service** folder to configure operations with security implications for this database mashable,
2. Clear or set any of the options to disable or enable specific operations for all the tables and views in this database mashable.
3. Click **Save**.

Viewing Details for Tables, Views or Stored Procedures

Once you have identified the datasource, schema or catalog, and specific database artifacts for the mashable you are registering, you can view more detailed information – the metadata – for each artifact. Simply select the artifact in the left pane to display metadata in the right pane.

Table, View or Stored Procedure Name

Detailed metadata displays a **Java** name which is the name that the MashZone NextGen mashable uses for the table, view or stored procedure. The **SQL** name is the database's name for this artifact. See "Database Mashable Names versus Database Names" on page 116 for more information.

Table or View Column Details

Tables and views show the following detailed information for their columns:

- **Name**: the MashZone NextGen mashable name for this column. See "Database Mashable Names versus Database Names" on page 116 for more information.
- **Java Type**: the Java class that the mashable uses as the datatype for this column.
- **SQL Type**: the database datatype for this column.
- **PK**: this is checked for any columns that are primary keys for tables. This is not applicable for views.

Table or View Finder Details

Tables and views show the following detailed information for their finders:

- **Name**: the operation name in the mashable for this finder.
- **Query Type**: the SQL keyword used to specify this finder. Typically finders use a WHERE clause.
- **Multi-finder**: whether this operation returns multiple rows.

Stored Procedure Details

Detailed information for stored procedures can vary based on what your database supports. It can include:

- For parameters:
  - **Name**: the name for this parameter.
  - **Java Type**: the Java datatype for this parameter.
  - **JDBC Type**: the JDBC datatype for this parameter.
  - **Direction**: IN, OUT or IN/OUT
- For the columns in the result set (if any):

- **Name**: the MashZone NextGen mashable name for this column.
- **Java Type**: the Java datatype for this column.
- **JDBC Type**: the JDBC datatype for this column.

Removing Tables, Views or Stored Procedures from the Mashable

As you review table, view or stored procedure details, you can remove them from the database mashable that you are registering.

1. Select the table, view or stored procedure from the folder in the left pane. Metadata for that database artifact displays in the right pane.
2. Click **Remove**.

Removing Columns for Table or View Operations

You can remove columns from the result set of finder operations for tables or views to simplify results or avoid permissions issues. You cannot remove columns that are primary keys.

Note: When you remove a column, the mashable definition for the table or view is changed. This affects all operations including queries and the insert and update operations for tables. MashZone NextGen also automatically removes any finders that are dependent on that column.

1. Select the table or view from the folder in the left pane.
2. Click **Configure** in the right pane.

   The Configure Columns and Finders window opens.
3. If needed, click the **Columns** tab.
4. Select a column in the left pane and click **Remove**.
5. Click **Close** once you have removed all unnecessary columns.

Changing Column Names or Datatypes for Tables or Views

Generally, you should **not** change the name or datatype information for columns. In rare cases, however, column datatypes may not be correctly mapped to JDBC datatypes or column names may use characters or keywords that MashZone NextGen does not support.

If this occurs, you must register the mashable again and manually correct the name or datatype information for the affected column.

1. Select the table or view from the folder in the left pane.
2. Click **Configure** in the right pane.

   The Configure Columns and Finders window opens.

3. If needed, click the **Columns** tab.
4. Select a column in the left pane.
5. Update the appropriate field for this column and click **Save**.
6. Click **Close** once you have removed all unnecessary columns.

Removing Table or View Finders

You can remove finders for tables or views to remove these operations in the mashable that you are registering.

1. Select the table or view from the folder in the left pane.
2. Click **Configure** in the right pane.

   The Configure Columns and Finders window opens.
3. Click the **Finders** tab.
4. Select a finder in the left pane and click **Remove**.
5. Click **Close** once you have removed all unnecessary finders.

Changing Finders for Tables or Views

You can also change the default finders that MashZone NextGen generates for tables or views or any finders that you add. You can change the finder name (the operation name for the mashable), add or change parameters, change the WHERE clause or add an ORDER BY clause to sort results.

1. Select the table or view from the folder in the left pane.
2. Click **Configure** in the right pane.

   The Configure Columns and Finders window opens.
3. Click the **Finders** tab.
4. Select the finder you want to change in the left pane.

   Detailed information for this finder displays in the right pane.
5. Double-click in the **Name**, **Where Clause** or **Order By** fields to change these values.

   Be sure to:

   - Use database names for columns in the **Where Clause** or **Order By** fields.
   - Identify parameters in the **Where Clause** in the form :*parameter-name* . For example:

     ```
     DEPARTMENT_ID = :dept
     ```
   - Make sure that all parameters used in the **Where Clause** are also defined in the Parameters table in this screen.

6. To add a parameter for use in the **Where Clause**, click .

   Double-click in the cells of the empty table row to define the parameter:

   - **Name**: the name for this parameter. This must be a legal SQL name.
   - **Java Type**: the Java class for the datatype of this parameter.
   - **JDBC Type**: the JDBC datatype of this parameter.
   - **Precision**: the JDBC precision for a numeric parameter.
   - **Scale**: the JDBC scale for a numeric parameter.
7. To delete a parameter, select it in the Parameters table and click .
8. Click **Save**.
9. Click **Close**.

Adding Finders for Tables or Views

You can define additional custom finders (query operations) for tables or views.

1. Select the table or view from the folder in the left pane.
2. Click **Configure** in the right pane.

   The Configure Columns and Finders window opens.
3. Click the **Finders** tab.
4. Click **New**.

   The fields you must complete for this finder displays in the right pane.
5. Enter a **Name** for this finder.

   The name must be a valid Java method name.
6. Enter the SQL code to select specific records in the **Where Clause**.

   - Use database names for columns.
   - Use valid SQL operators, such as = or `LIKE`.
   - Identify parameters in the form `:`*mashable-column-name* . For example:

     ```
     DEPARTMENT_ID = :departmentId
     ```
   - Make sure that all parameters are also defined in the Parameters table in this screen.
7. If results should be sorted, enter a list of columns, in the **Order By** field using database column names separated by commas.
8. To add a parameter for use in the **Where Clause**, click .

   Double-click in the cells of the empty table row to define the parameter:

- **Name**: the mashable name for the column for this parameter.
- **Java Type**: the Java class for the datatype of this parameter.
- **JDBC Type**: the JDBC datatype of this parameter.
- **Precision**: the JDBC precision for a numeric parameter.
- **Scale**: the JDBC scale for a numeric parameter.

9. Click **Save**.
10. Click **Close**.

Register SharePoint Lists

You can register SharePoint lists or specific list views as mashable information sources in MashZone NextGen*only* if the MashZone NextGen Add-On for SharePoint is installed and configured by your MashZone NextGen administrator. .

> **Note:** If the MashZone NextGen Add-On for SharePoint is installed and configured, users can also register SharePoint lists as mashable information sources in MashZone NextGen from SharePoint.

Once registered, you can then use information from SharePoint list mashables in mashups or apps.

1. Select **Connect > SharePoint List** in hte MashZone NextGen Hub main menu.
2. Enter a unique **Name** for this mashable information source.

   MashZone NextGen uses the mashable name to assign a unique identifer to mashables. Mashable names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

   It is a good practice to name mashables based on their purpose, such as `AccountsPayable`, rather than on deployment or technical information. This will make it easier for other users to recognize and work with the mashable.
3. Choose the **Connection** for the SharePoint site that hosts the list you want to use as an information source.

   MashZone NextGen retrieves information for this SharePoint connection and retrieves the default credentials for this connection (in User Credentials).

   > **Note:** If no SharePoint connections have been configured in MashZone NextGen, this field has an error message. You cannot register mashables from SharePoint until a connection has been configured by a MashZone NextGen administrator.
   >
   > If the SharePoint connection you need to work with is not in this list, contact your MashZone NextGen administrator.

4. If needed, click **User Credentials for *connection*** to change the credentials (username, password and domain) to use with this connection for this mashable. Click **Update**.

   Important: Updating these credentials creates or updates *your*MashZone NextGen User Attributes for this SharePoint connection. This can affect how other mashups or mashables that work with SharePoint behave for you if they are configured to use the current user's credentials.

   The permissions for the credentials you use determine which SharePoint sites and lists are visible for registration. Based on configuration:

   - This may default to the credentials defined for the connection you have selected, also known as the *service account*.
   - Or you may be required to supply your credentials for this SharePoint connection.
   - Or the connection may disable the use of user credentials based on configuration and not allow updates.

   To remove your credentials and return these to the default credentials for this SharePoint connection, click **Remove**. This also *removes* your MashZone NextGen User Attributes for this SharePoint server.

   If you update these credentials, MashZone NextGen uses them to get new site and list information based on the permissions for your user account.

5. Choose the **Site Name** and the **List Name** for the list you want to use.
6. If desired, choose the specific **View** of this list that you want to use.
7. If needed, click **Request Options** to control which user credentials are used to access SharePoint when this mashable is run.

   By default, the SharePoint credentials for the user who is running the mashable are used to access SharePoint. These credentials are defined as MashZone NextGen User Attributes for each SharePoint server in the user's MashZone NextGen Hub profile.

   - **Bind now** = set this option to always use the credentials you have supplied while registering this SharePoint list as a mashable.
   - **Bind at runtime dynamically** = this is set by default. With this option, access to SharePoint uses the credentials that users have added to their MashZone NextGen Hub profile, rather than the credentials you are supplying to register this mashable.

8. Add metadata for this mashable, such as a description or category. For more information, see "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 external.
9. Click **Register**.

   MashZone NextGen registers the mashable and turns this mashable information source on (by default).

Then, either:

- Grant permissions to run this mashable to other users and groups:
  - Choose the mashble-wide **Service >> *mashable name*** permissions section to grant run permissions to all operations. Or expand a specific **Operation >> *operation name*** section to grant run permissions for just that operation.
  - "Find Users and Groups for Permissions" on page 104 or "Choose MashZone NextGen Built-In Groups for Permissions" on page 105.
  - Click on a user or group in the **Search result** list to add them to the list of **Principals with permissions**.
  - Save the permissions once the list is complete.
- Or skip permissions for now and click **Finish**.

  To allow other users to work with this mashable, you must grant run permissions at a later time.

Note: Some groups or users may already be listed as **Principals with permissions**. These users and groups have been defined by a MashZone NextGen administrator to grant permission automatically.

You can open the artifact page for this new mashable to run the mashable, add views, take snapshots, create basic apps based on this mashable, edit the mashable, turn it off or grant permissions to other users to work with it.

Register a Snapshot as a Mashable

To share snapshots with other users, easily use a snapshot in a mashup in Wires or create a basic app from a snapshot, you must register it as a mashable. MashZone NextGen developers can use snapshots directly in mashups or custom apps with the MashZone NextGen Snapshot API.

Note: The snapshot feature is only available if your MashZone NextGen administrator has enabled this feature.

1. Open the mashable or mashup for this snapshot and view the snapshot. See "Find and View Snapshots" on page 183 for instructions.
2. Click **Make Mashable**.
3. The name and description for the mashable default to the name and description for the snapshot. Change these fields if needed and click **Register**.

Subscribe to Events from the Event Bus or Apama

Events from other Software AG applications, such as Optimize and Apama, can be used as mashable information sources. You connect MashZone NextGento the Event Bus or

directly Apama, in some cases, and subscribe to a specific type of event or scenario. This subscription creates an event mashable in MashZone NextGen.

MashZone NextGen administrators create the connections and subscriptions for event mashables.

With Apama events, MashZone NextGen can directly accesses event data in two ways:

- *Scenarios* in Apama have event data that has been transformed specifically for use in dashboards. (This is also sometimes called *dataviews*.)

  MashZone NextGen works with Apama scenario events through the Event Service and event sources defined by a MashZone NextGen administrator.

- *Distributed Stores* in the Apama MemoryStore hold data that can be shared across several correlators and monitors and can be used in dashboards. In some cases, this may be a very large dataset.

  Data from these distributed Apama stores cannot be accessed as scenarios or as events. If the Apama MemoryStore uses BigMemory, however, MashZone NextGen mashups can access this data using MashZone NextGen Analytics and *dynamic external stores*. See "Example 129. Load Datasets from a Dynamic External In-Memory Store" on page 872 for an example of mashup access.

Create or Edit MashZone Feeds

MashZone feeds access many common types of data sources, such as databases or CSV files, as well as business or other types of events that are routed from the Event Bus. They can also convert or transform data much like mashups.

You create or edit MashZone feeds in the MashZone Feed Editor. To get started, see "Edit an Existing MashZone Feed" on page 159 or "Create a New MashZone Feed" on page 160.

Edit an Existing MashZone Feed

MashZone feeds are added to MashZone NextGen when users create them, if creation is enabled, or when MashZone NextGen administrators import these feeds from an existing MashZone Server. To edit a MashZone feed:

1. Find the MashZone feed you want to edit in search results, from favorites or another link. See "Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub" on page 84 for more information.
2. Open the MashZone feed.
3. In the feed's artifact page, select **Edit > Edit with Editor**

   The MashZone Feed Editor opens with this feed loaded.
4. See the following topics in the MashZone documentation for information on using the MashZone Feed Editor and links to editing instructions:

   - *MashZone Feed Editor Features*

- *MashZone Feed Definitions*

5. Once you have saved your changes, you may also need to:
   - Grant permissions to run this MashZone feed to other users. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.
   - Add one or more views to this MashZone feed to define different ways to see this data. See "Add Views to Mashables and Mashups" on page 901 for instructions.
   - Create a basic app with this MashZone feed or add the feed to a workspace app. See "Create a Basic App" on page 1093 or "Create Workspace Apps with Mashboard" on page 1116 for instructions.

Create a New MashZone Feed

Note: Creation of new MashZone feeds may be disabled by MashZone NextGen administrators. MashZone NextGen administrators should see *Disable the MashZone Feed Editor for New Feeds* in the MashZone documentation for more information.

1. Select **Mashups > MashZone Feed Editor** from the main menu.

   The MashZone Feed Editor opens with the Output element that completes the feed definition already created.

2. To define the feed, you:
   - Add one or more data sources.
   - View data source results, if needed, to determine what conversions or transformations are needed.
   - Add operators to convert, calculate or transform data.
   - If needed, add user input parameters to allow users to select the data they want to see.
   - Link the final results to the output to complete the feed definition.
   - Save the feed definition.

   See the MashZone documentation topics *Create a MashZone data feed* for an example illustrating the basic steps and the *MashZone Feed Editor Features* topic for basic information on working in this editor.

   For more information, see *Use data sources*, *Use operators* and *Use user inputs* in the MashZone documentation.

3. Once you have saved the MashZone feed, you should open the artifact page for this feed and:
   - Grant permissions to run this MashZone feed to other users. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.

- Add one or more views to this MashZone feed to define different ways to see this data. See "Add Views to Mashables and Mashups" on page 901 for instructions.
- Create a basic app with this MashZone feed or add the feed to a workspace app. See "Create a Basic App" on page 1093 or "Create Workspace Apps with Mashboard" on page 1116 for instructions.

Mashable Authentication with Security Profiles

Security profiles define the credentials or other security information needed to securely connect to a mashable information source and be authenticated. The authentication information that you provide when you register a mashable is the default authentication that MashZone NextGen sends to the web service when the mashable is invoked.

Note: Passwords provided as part of credentials for a security profile are encrypted in the MashZone NextGen Repository.

You can provide authentication information as direct values when you register a mashable. Or this information can be dynamic, using MashZone NextGen global or user attributes. See "Using MashZone NextGen Global or User Attributes in Security Profiles" on page 163 for more information on dynamic values.

The security information needed for a security profile depends on the secure connection protocol required by the mashable information source. MashZone NextGen provides the following security profiles that you can use with mashables:

- HTTP Basic Authentication
- CAS2 Authentication
- NTLM Authentication
- SSL Authentication
- SSO Authentication

Other custom security profiles may also be available for you to use. See "Custom Security Profiles" on page 163 for more information.

HTTP Basic Authentication

This security profile sends a username and password in the HTTP Authentication header in requests to a mashable.

Enter the default **Username** and **Password** to access this mashable. Requests to invoke the mashable can override these default credentials.

CAS2 Authentication

This security profile is used for mashables secured by the Central Authentication Service (CAS) single sign-on solution using the CAS2 protocol. With CAS2, MashZone NextGen acts as a proxy to pass the CAS ticket for the current user to the mashable endpoint.

Note: CAS2 security profiles are only available if MashZone NextGen is configured to use CAS for authentication.

In most cases, no configuration is needed in a CAS2 security profile. If the mashable endpoint uses a URL for ticket validation that is different from the mashable endpoint URL, however, you must enter the ticket validation URL for the mashable as the **casServiceUrl**.

NTLM Authentication

This security profile is used for mashables in Windows environments. It is based on Windows NT Domains.

Enter the default **Username**, **Password** and **Domain** to access this mashable. Requests to invoke the mashable can override these default credentials.

SSL Authentication

This security profile uses mutual SSL to exchange digital certificates for mashables that use the HTTPS protocol in their URL and require a certificate from the MashZone NextGen Server for authentication.

Note: Do *not* use this security profile for mashables that use one-way SSL.

Before you register a mashable with this security profile, you or a MashZone NextGen administrator must:

- Configure MashZone NextGen for SSL.
- Obtain the digital certificate for the mashable information source, *if the certificate is self-signed*, and add this certificate to the MashZone NextGen trust store.

To enable this mutual exchange of digital certificates, you complete two properties:

- **TrustedCertificateName**: you only need to set this property if the certificate for this mashable is self-signed. Certificates for mashables that are signed by well-known Certificate Authorities are automatically verified and added to the Trust Store.

  Open the list for this property and select the alias that was used to import the self-signed digital certificate for this mashable information source.
- **PrivateCertificateName**: open the list for this property and select the alias for the digital certificate for MashZone NextGen. This certificate supplies the authentication information required to securely connect to this mashable information source.

Contact your MashZone NextGen administrator for information on the appropriate alias names to choose.

SSO Authentication

This authentication profile uses credentials or other authentication provided by a single sign-on (SSO) solution used by your organization.

Note: This security profile is appropriate with agent-based SSO solutions, such as Netegrity SiteMinder. See the CAS2 security profile when SSO is handled by CAS.

With this profile, the SSO solution authenticates users and adds a authentication token in a well-known, HTTP header, that the user's browser includes in requests to MashZone NextGen, services or other applications. MashZone NextGen uses this token to verify authentication with the SSO solution when users work in MashZone NextGen Hub or the AppDepot. MashZone NextGen also forwards this HTTP header to mashable information sources with this security profile when they are invoked.

Enter the name of the HTTP header that contains user authentication tokens for your SSO solution in the **Ssotoken** field. The **Ssotokensource** field is not currently used.

Custom Security Profiles

MashZone NextGen developers can create custom security profiles for mashables that have other security requirements not handled by the built-in profiles in MashZone NextGen. MashZone NextGen administrators must register these custom security profiles to make them available when you register mashables.

Developers and administrators should see "Configure Secure Connections for Mashables" on page 164 for more information.

Using MashZone NextGen Global or User Attributes in Security Profiles

In some cases, the authentication information you need to send to a mashable must be specific to each user when they work with that mashable. In other cases, only MashZone NextGen administrators have the correct authentication information to use for a mashable.

In either of these cases, you can use MashZone NextGen global or user attributes to supply the information:

1. MashZone NextGen administrators or users must first create a MashZone NextGen attribute.
   - *MashZone NextGenGlobal Attributes*: MashZone NextGen administrators create these attributes in the Admin Console. .
   - *MashZone NextGenUser Attributes*: each user that will work with this mashable creates their own attribute *using the same attribute name* in their MashZone

NextGen user profile. See "Manage Your MashZone NextGen User Attributes" on page 82 for instructions.

2. In any field for a security profile, type `$` to open a list of MashZone NextGen global and user attributes to choose from.

Configure Secure Connections for Mashables

Mashable information sources may require credentials, specific secure protocols or other security information to accept connections from MashZone NextGen and process requests. These security requirements are called *security profiles* in MashZone NextGen.

Users provide credentials or other security information when they register mashables with MashZone NextGen based on the security profile they choose during registration. MashZone NextGen provides security profiles for some of the most common secure connection requirements:

- *HTTP Basic authentication* = basic credentials (user name and password) for authentication with the mashable information source.
- *CAS2* = authentication is handled by the Central Authentication Server (CAS), a single sign-on solution, using the CAS2 protocol.

  To enable this security profile, MashZone NextGen administrators *must* also configure MashZone NextGen authentication for SSO using CAS2.
- *NTLM authentication* = Windows credentials (user name, password and NT domain) for authentication with the mashable information source.
- *SSL and Digital Certificates* = HTTPS using mutual SSL for a secure connection and the certificate to provide authentication with the mashable information source.

  Tp enable this security profile, MashZone NextGen administrators *must* also configure MashZone NextGen for SSL.
- *SSO* = credentials or other authentication information provided by user-agent-based single sign-on solutions, such as SiteMinder.

  To enable this security profile, MashZone NextGen administrators *must* also configure MashZone NextGen authentication for SSO.

If a mashable has other security requirements, MashZone NextGen developers or administrators can "Implement a Custom Security Profile Client" on page 164 that handles these unique requirements and then "Register a Custom Security Profile" on page 171 with MashZone NextGen.

Implement a Custom Security Profile Client

Custom security profiles use HTTP or HTTPS to connect to the mashable. They are implemented in Java, using the MashZone NextGen Security Profile API and consist of these components:

- *Security Profile* = an object with the credentials or other security information that users provide when they register a mashable.
- *Service Invocation Context* = context information for the specific request to MashZone NextGen to invoke this mashable.
- *Security Profile Client* = the client class that handles making secure connections, constructing credentials or other security information and sending requests with this security information to mashables that use this type of profile.

When mashables are invoked, MashZone NextGen instantiates both the security profile and the service invocation context based on saved metadata from the MashZone NextGen Repository and the request to MashZone NextGen to invoke the mashable. To implement the security profile client, you:

1. Add the following JAR file to your classpath to include the Security Profile API in your development environment:

   web-apps-home /presto/WEB-INF/lib/presto-common.jar

2. Import API Classes and Extend BaseSecureServiceClient
3. Implement the getData Method
4. Implement the Remaining SecureServiceClient Methods

Once you have implemented the custom security profile client, a MashZone NextGen administrator must Register a Custom Security Profile in MashZone NextGen to allow users to select this profile when they register mashables.

Import API Classes and Extend BaseSecureServiceClient

In your custom security profile client class, import the following classes:

```
package com.myorg.security.service.profile
...
//import com.jackbe.jbp.common.httpclient.SecureServiceClient;
import com.jackbe.jbp.common.httpclient.BaseSecureServiceClient;
import com.jackbe.jbp.common.httpclient.SecureServiceClientException;
import com.jackbe.jbp.common.plugin.ServiceInvocationContext;
import com.jackbe.jbp.common.security.profile.SecurityProfile;
...
```

Note: `BaseSecureServiceClient` implements the `SecureServiceClient` interface, so only one of these import statements is needed.

Have your client class extend `BaseSecureServiceClient` or implement the `SecureServiceClient` interface:

```
package com.myorg.security.service.profile
...
//import com.jackbe.jbp.common.httpclient.SecureServiceClient;
import com.jackbe.jbp.common.httpclient.BaseSecureServiceClient;
import com.jackbe.jbp.common.httpclient.SecureServiceClientException;
import com.jackbe.jbp.common.plugin.ServiceInvocationContext;
import com.jackbe.jbp.common.security.profile.SecurityProfile;
...
public class MyCustomSecureClient extends BaseSecureServiceClient {
  ...
```

```
}
```

The `BaseSecureServiceClient` class includes convenience methods to:

- Get the MashZone NextGen default HTTP client (Apache)
- Use the configured proxy server, if any, for MashZone NextGen
- Convert HTTP headers from the mashable response to a Map which allows them to be easily forwarded in the MashZone NextGen response

Implement the getData Method

You must implement the `getData(java.lang.String requestData, ServiceInvocationContext invCtx)` method to actually invoke the mashable. This method is also responsible for using the credentials or other security information from the security profile to build a request in the form the mashable expects and to handle any other security requirements for the connection to the mashable.

The `requestData` input parameter for this method supplies the body of the request that MashZone NextGen has built to send to this mashable. The `invCtx` parameter is the `ServiceInvocationContext` instance for this invocation. This also contains the `SecurityProfile` for this mashable.

A very common flow for `getData` is to:

1. Get an HTTP client using the `getDefaultHTTPClient` method.

```
public InputStream getData(String requestData, ServiceInvocationContext invCtx) throws SecureS
  String username="";
  String password="";
  String token="";
  logger.debug("...invoking using SampleSecureServiceClient...");
  try {
    httpClient = getDefaultHTTPClient();
    ...
  } catch(HttpException ex) {
    logger.debug("http error...", ex);
    throw new SecureServiceClientException("http error", ex);
  } catch(IOException ex) {
    logger.debug("io error...", ex);
    throw new SecureServiceClientException("io error", ex);
  } catch(Exception ex) {
    logger.debug("error...", ex);
    throw new SecureServiceClientException("error", ex);
  }
}
```

2. Get the security parameter values that were defined when the mashable was registered from the `SecurityProfile` in `ServiceInvocationContext`:

```
...
  try {
    httpClient = getDefaultHTTPClient();
    //get parameters from security profile
    SecurityProfile secProfile = invCtx.getSecurityProfile();
    if(secProfile!=null) {
      Map profileParams = secProfile.getParams();
      for(Iterator itr=profileParams.entrySet().iterator(); itr.hasNext();) {
        Map.Entry entry = (Map.Entry<String, String>)itr.next();
        if (((String)entry.getKey()).equals("username")) {
```

```
          username = (String)entry.getValue();
        }
        else if (((String)entry.getKey()).equals("pwd")) {
          password = (String)entry.getValue();
        }
        else if (((String)entry.getKey()).equals("token")) {
          token = (String)entry.getValue();
        }
      }
    }
  ...
  }
}
```

3. Then build the credentials and other security information into the form expected by the mashable and add this to the message.

```
...
  try {
    httpClient = getDefaultHTTPClient();
    //get parameters from security profile
    SecurityProfile secProfile = invCtx.getSecurityProfile();
    if(secProfile!=null) {
      Map profileParams = secProfile.getParams();
      for(Iterator itr=profileParams.entrySet().iterator(); itr.hasNext();) {
        Map.Entry entry = (Map.Entry<String, String>)itr.next();
        if (((String)entry.getKey()).equals("username")) {
          username = (String)entry.getValue();
        }
        else if (((String)entry.getKey()).equals("pwd")) {
          password = (String)entry.getValue();
        }
        else if (((String)entry.getKey()).equals("token")) {
          token = (String)entry.getValue();
        }
      }
    }
    if (token==null || token.trim().length()=0) {
      logger.error("no token in security profile");
    } else {
     //build credentials in correct form and add to request
    }
...
  }
}
```

4. Determine the HTTP method to use for this request. This is based on the *invocation type* in the `ServiceInvocationContext`:

```
...
    if (token==null || token.trim().length()=0) {
      logger.error("no token in security profile");
    } else {
     //build credentials in correct form and add to request
    }
    //get HTTP method based on invocation type
    String invType = invCtx.getInvocationType();
    if(invType==null || invType.trim().length()==0)
      invType = ServiceInvocationContext.INVOCATION_TYPE_GET;
    if(invType.equals(ServiceInvocationContext.INVOCATION_TYPE_GET)) {
        httpMethod = new GetMethod(requestData);
    } else if(invCtx.getInvocationType().equals(ServiceInvocationContext.INVOCATION_TYPE_POST)
      httpMethod = new PostMethod(invCtx.getServiceURL());
      ((PostMethod)httpMethod).setRequestBody(requestData);
```

```
    }
...
  }
}
```

5. Make sure the client can use the proxy server if one has been configured for MashZone NextGen. This uses the `setupHttpProxy` convenience method from `BaseSecureServiceClient`:

```
...
    //get HTTP method based on invocation type
    String invType = invCtx.getInvocationType();
    if(invType==null || invType.trim().length()==0)
      invType = ServiceInvocationContext.INVOCATION_TYPE_GET;
    if(invType.equals(ServiceInvocationContext.INVOCATION_TYPE_GET)) {
        httpMethod = new GetMethod(requestData);
    } else if(invCtx.getInvocationType().equals(ServiceInvocationContext.INVOCATION_TYPE_POST)
      httpMethod = new PostMethod(invCtx.getServiceURL());
      ((PostMethod)httpMethod).setRequestBody(requestData);
    }
    // setup http proxy if required
    setupHttpProxy(httpClient, httpMethod);
...
  } catch(HttpException ex) {
    logger.debug("http error...", ex);
    throw new SecureServiceClientException("http error", ex);
  } catch(IOException ex) {
    logger.debug("io error...", ex);
    throw new SecureServiceClientException("io error", ex);
  } catch(Exception ex) {
    logger.debug("error...", ex);
    throw new SecureServiceClientException("error", ex);
  }
}
```

6. Invoke the mashable and return the response:

```
...
    // setup http proxy if required
    setupHttpProxy(httpClient, httpMethod);
    // invoke
    httpClient.executeMethod(httpMethod);
    // return data as stream
    return httpMethod.getResponseBodyAsStream();
  } catch(HttpException ex) {
    logger.debug("http error...", ex);
    throw new SecureServiceClientException("http error", ex);
  } catch(IOException ex) {
    logger.debug("io error...", ex);
    throw new SecureServiceClientException("io error", ex);
  } catch(Exception ex) {
    logger.debug("error...", ex);
    throw new SecureServiceClientException("error", ex);
  }
}
```

The following example, however, shows `getData` that does not use the default HTTP client. This example is for a security profile for WSDL mashables that require a WSS Policy using the Oasis User Token Profile. It gets the security profile from the `invCtx` parameter and uses this data to build the SOAP header for the WSS Policy. It then constructs the SOAP message using the SOAP body passed in `requestData`.

```
  @Override
```

```
public InputStream getData(String soapBodyStr, ServiceInvocationContext invCtx) throws SecureSe
{
  String username = "";
  String password = "";
  String soapAction = "";
  String endpoint = "";
  SoapVersion soapVersion = SoapVersion.Soap11;
  //get security profile data
  SecurityProfile secProfile = invCtx.getSecurityProfile();
  if(secProfile!=null) {
    Map<String,String> profileParams = secProfile.getParams();
    for(Iterator itr=profileParams.entrySet().iterator(); itr.hasNext();) {
      Map.Entry entry = (Map.Entry<String, String>)itr.next();
      if (((String)entry.getKey()).equals("username")) {
        username = (String)entry.getValue();
      }
      else if (((String)entry.getKey()).equals("pwd")) {
        password = (String)entry.getValue();
      }
      else if (((String)entry.getKey()).equals("soapAction")) {
        soapAction = (String)entry.getValue();
      }
      else if (((String)entry.getKey()).equals("soapVersion")) {
        if (((String)entry.getValue()).equals("1.1"))
          soapVersion = SoapVersion.Soap11;
        else if (((String)entry.getValue()).equals("1.2"))
          soapVersion = SoapVersion.Soap12;
      }
      else if (((String)entry.getKey()).equals("endpoint")) {
        endpoint = (String)entry.getValue();
      }
    }
  }
  byte[] nonceValue = null;
  try {
    nonceValue = WSSecurityUtil.generateNonce(16);
  }
  catch (WSSecurityException ex) {
  }
  DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
  dateFormat.setTimeZone(TimeZone.getTimeZone("UTC"));
  String soapNamespace = "";
  try {
    XmlObject soapBody = XmlObject.Factory.parse(soapBodyStr);
    XmlCursor cursor = soapBody.newCursor();
    if (soapVersion == SoapVersion.Soap12) {
      soapNamespace = SOAP_12_NS;
      cursor.selectPath("declare namespace soap='" + soapNamespace + "';./soap:Envelope/soap:Bo
    }
    else if (soapVersion == SoapVersion.Soap11) {
      soapNamespace = SOAP_11_NS;
      cursor.selectPath("declare namespace soap='" + soapNamespace + "';./soap:Envelope/soap:Bo
    }
    else {
       logger.error("OASIS User-Token-Profile 1.0, invalid soap version set to " + soapVersion)
    }
    //build SOAP header for WSS policy token
    String UTPSOAPHeader =
      "<soap:Header xmlns:soap=\"" + soapNamespace + "\">" +
        "<wsse:Security soapenv:mustUnderstand=\"1\"
        xmlns:wsse=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
          "<wsse:UsernameToken wsu:Id=\"UsernameToken-1\"
        xmlns:wsu=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
```

```
              "<wsse:Username>" + username + "</wsse:Username>" +
              "<wsse:Password Type=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-user
              + password + "</wsse:Password>" +
              "<wsse:Nonce EncodingType=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
              + Base64.encode(nonceValue) + "</wsse:Nonce>" +
              "<wsu:Created>" + dateFormat.format(new Date()) + "</wsu:Created>" +
            "</wsse:UsernameToken>" +
          "</wsse:Security>" +
        "</soap:Header>";
      XmlObject soapHeader = XmlObject.Factory.parse(UTPSOAPHeader);
      XmlObject inputMessage = null;
      if (cursor.hasNextSelection() && cursor.toNextSelection()) {
        do {
          cursor.toNextToken();
        }
        while(!cursor.isContainer());
          inputMessage = XmlObject.Factory.parse(cursor.xmlText());
      }
      //build full SOAP request
      XmlObject soapRequest = SoapMessageGenerator.buildSoapRequest(false, soapHeader, inputMessa
      WSSecUsernameToken usernameToken = new WSSecUsernameToken();
      SoapClient soapClient = new SoapClient();
      if (soapAction != null) {
        if (soapAction.length() == 0)
          soapClient.setSoapAction("\"\"");
        else
          soapClient.setSoapAction("\"" + soapAction + "\"");
      }
      if (endpoint != null && endpoint.length() > 0)
        soapClient.setEndpointRef(endpoint);
      else
        soapClient.setEndpointRef(invCtx.getServiceURL());
      //invoke service
      soapClient.setSoapVersion(soapVersion);
      soapClient.loadSoapRequestFromXmlObject(soapRequest);
      System.out.println(soapRequest.xmlText());
      XmlObject responseXml = soapClient.invoke();
      return WsdlUtils.fromString(responseXml.xmlText());
    }
    catch(XmlException ex) {
       logger.error("fail parsing soap header or body: " + ex.getMessage(), ex);
    }
    catch(SoapClientException ex) {
      logger.error("fail invoking the service: " + ex.getMessage(), ex);
    }
    return null;
  }
```

Implement the Remaining SecureServiceClient Methods

You must implement the remaining `SecureServiceClient` methods, although they may not perform any operations depending on the requirements for this security profile:

- `getMetaData(java.lang.String requestData, ServiceInvocationContext invCtx)`: this method is used when mashable information sources are registered with this security profile. Typically, this also invokes the service, just as `getData` does. For example:

  ```
  public InputStream getMetadata(String requestData, ServiceInvocationContext invCtx) throws Se
    // no unique requirements, just call getData
    return getData(requestData, invCtx);
  }
  ```

- `getResponseHeaders()`: to retrieve HTTP headers from the mashable's response. This is most commonly used to allow headers to be included in the MashZone NextGen response. For example:

```
public Map<String,String> getResponseHeaders() {
  Header[] headers = httpMethod.getResponseHeaders();
  Map<String,String> responseHeaders = convertHeadersToMap(headers);
  return responseHeaders;
}
```

  This example uses the `convertHeadersToMap` convenience method from `BaseSecureServiceClient` to convert the array of HTTP headers to a Map so that MashZone NextGen can include them in the response.

- `getStatusCode()`: to retrieve the HTTP status code from the mashable's response. For example:

```
public int getStatusCode() {
  return httpMethod.getStatusCode();
}
```

- `getStatusText()`: to retrieve the HTTP status text from the mashable's response. For example:

```
public String getStatusText() {
  return httpMethod.getStatusText();
}
```

- `close()`: to explicitly close the connection. For example:

```
public void close() {
  if(httpMethod!=null)
    httpMethod.releaseConnection();
}
```

Register a Custom Security Profile

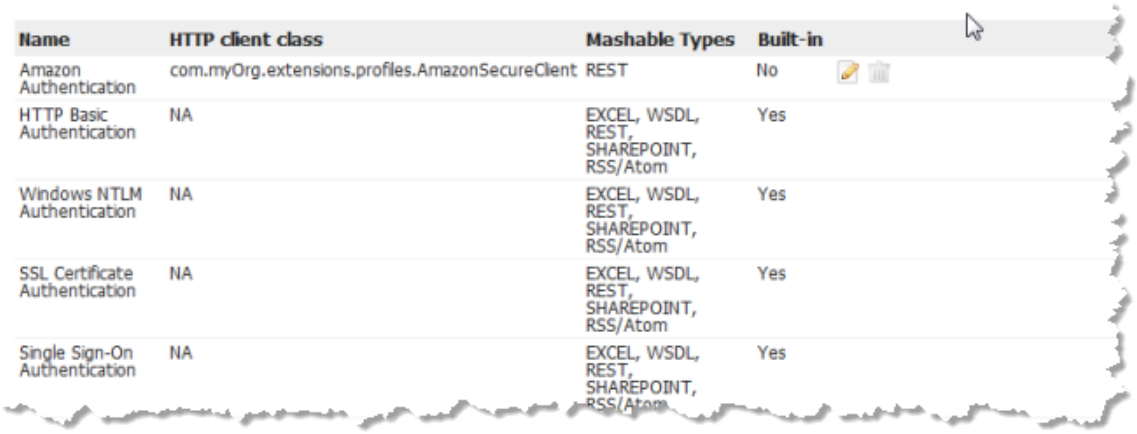
Once you have implemented a security profile client, compiled it and tested it, you must deploy this in MashZone NextGen and then register the security profile to make this profile available to users when they register mashables.

1. Deploy your custom security profile client class, individually or in a JAR, to the MashZone NextGen Server in one of these locations:
   - The external configuration folder, if any, for the MashZone NextGen Server.

     Important: Deploying extensions to MashZone NextGen in an external configuration folder simplifies future deployments or management of MashZone NextGen Server clusters.

   - *web-apps-home* /presto/WEB-INF/classes for individual classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
   - *web-apps-home* /presto/WEB-INF/lib for a JAR file. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
2. Click Admin Console in the MashZone NextGen Hub main menu.

3. Expand the Security and Policies section and click **Security Profiles**. A list of all registered security profiles appears:

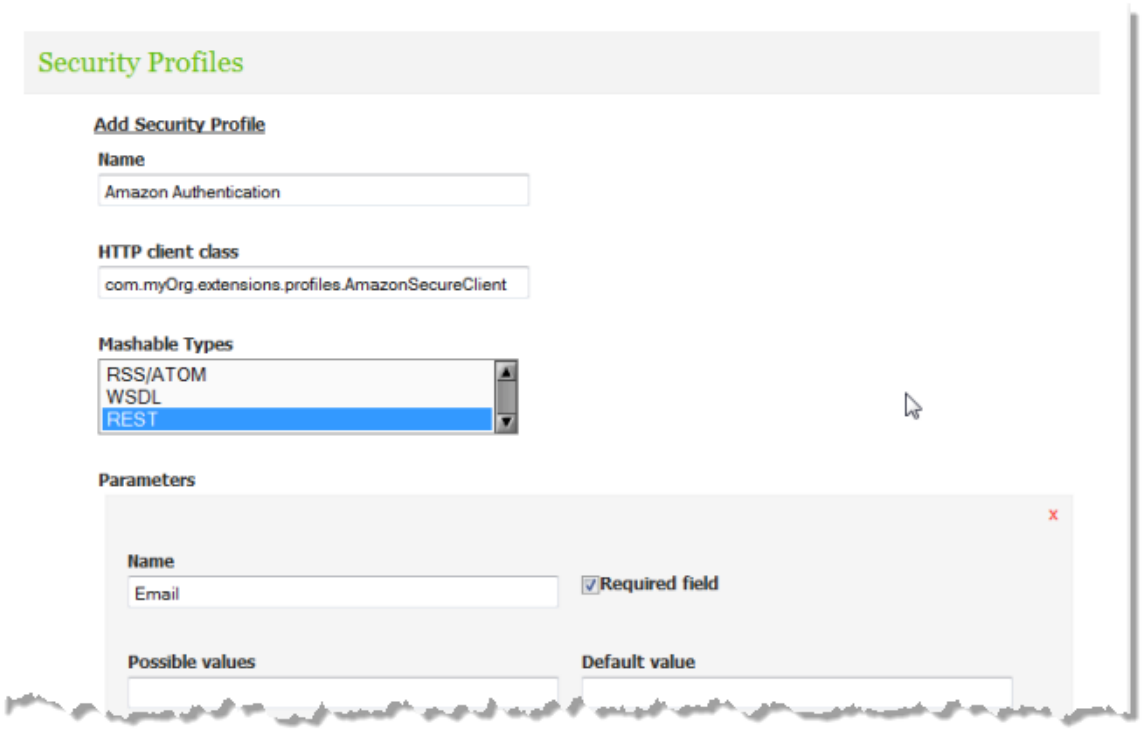


Name	HTTP client class	Mashable Types	Built-in
Amazon Authentication	com.myOrg.extensions.profiles.AmazonSecureClient	REST	No
HTTP Basic Authentication	NA	EXCEL, WSDL, REST, SHAREPOINT, RSS/Atom	Yes
Windows NTLM Authentication	NA	EXCEL, WSDL, REST, SHAREPOINT, RSS/Atom	Yes
SSL Certificate Authentication	NA	EXCEL, WSDL, REST, SHAREPOINT, RSS/Atom	Yes
Single Sign-On Authentication	NA	EXCEL, WSDL, REST, SHAREPOINT, RSS/Atom	Yes

   You cannot edit or delete the MashZone NextGen built-in profiles.

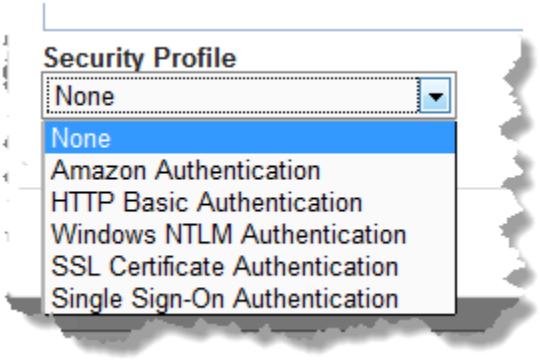
4. Click **Add new Security Profile**.
5. Enter a **Name** for this profile. This is the name that users will select when they register mashables.
6. Enter the fully qualified class name for your custom security profile client class (`SecureServiceClient` implementation or `BaseSecureServiceClient` extension) as the **HTTP client class**. For example:

   ```
   com.myOrg.extensions.profiles.myCustomSecureServiceClient
   ```



7. Choose all the types of mashable information sources that can use a security profile of this type.

   When users register mashables, this profile will appear only for mashables of these types.



8. If users must supply credentials or other security information for a profile of this type, follow these steps for each security property:

   a. Click **Add a parameter**.

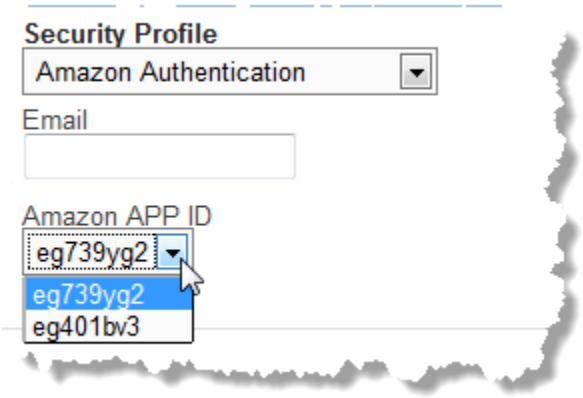
   b. Enter a **Name** for the parameter. This is the label that users will see when they register mashables with this type of security profile.

c. Set the **Required field** option if users must provide this information in all security profiles of this type.

d. Optionally, enter a list of **Possible values** that users should select from for this property. Separate values with commas.

e. Optionally, enter a **Default value** for this property.

When users register a mashable and choose this security profile, fields appear for each security property you add:



9. Click **Save this security profile**.

Configure HTTP Request Headers

This topic applies to MashZone NextGen 3.9 and higher.

REST, or *REpresentational STate web services*, are web services available via HTTP or HTTPS that use several HTTP methods to invoke service operations and pass input parameters.

See "REST Web Services" on page 129 for more information.

If you register a new REST Web Service in the MashZone NextGen Hub then you can specify HTTP Request Headers optionally.

See "Register REST Web Services" on page 138

After the Registration of a REST web wervice you can still configure the HTTP Request Headers in the MashZone NextGen Hub.

To configure HTTP Request Headers please proceed as follows:

1. Open the details page of the wanted REST web service in MashZone NextGen Hub from search results, bookmarks, favorites, or other links. See "Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub" on page 84.
2. Select **Show > Info**.

3. To configure the **HTTP Request Headers** click the **Edit** button on the info page.
4. Click the + button and fill in the required `Name` and `Value` parameters. You can add one or more HTTP Request Headers as Name:Value pairs. The **Name** field provides a list of HTTP headers as a drop-down menu, but also allows you to type in a valid name. Depending on the selected parameter, the **Value** field provides a list of appropriate values as a drop-down menu or you can type in a valid value.

   See **List of HTTP header fields on Wikipedia.org**
5. Click **Save**.
6. Close the info page.

REST Web Services

REST, or *representational state web services*, are web services available via HTTP or HTTPS that use URLs or the body of POST requests to invoke service operations and pass input parameters. REST web services may use either the GET or POST operations in HTTP.

Note: You can also register syndicated feeds (RSS or Atom) as REST web services.

MashZone NextGen uses the following operation names for REST web services based on the request method of the service:

- `getData`: for GET requests.
- `postData`: for POST requests.

Like several other types of mashables, MashZone NextGen supports secure communication with REST web services with these security profiles:

- Basic HTTP Authentication
- NTLM Authentication
- SSL

MashZone NextGen administrators can also add custom security mechanisms to handle specific information source security requirements. See "Configure Secure Connections for Mashables" on page 164 for more information.

MashZone NextGen administrators can also control the HTTP response headers for REST web services or timeouts for connections with REST web services.

You can also configure the HTTP request header for REST web services. See "Configure HTTP Request Headers" on page 174 for more information.

For information on registering, see "Register REST Web Services" on page 138.

GET-Based REST Web Services

Most REST web services accept requests that use the GET operation in HTTP to invoke the service. Parameters are passed within the URL to the service following the question mark (?) delimiter. The URL for a REST web service using GET might look like this:

```
http://www.someOrg.com/myRestService?date=20080301&zip=94102
```

When you register REST web services in MashZone NextGen, you provide the URL to the web service. For GET-based REST services, the URL should include any parameters that you need to use for any request. You can also include default values for these parameters or simply omit the value.

Note: MashZone NextGen uses the default parameter values that you supply when you register a REST web service to validate the URL during registering. MashZone NextGen*also* uses the default parameter values when the request to run the service contains *no parameters* at all.

POST-Based REST Web Services

Some REST web services, however, use the POST operation in HTTP to invoke the service. Parameters are passed in the body of the POST request.

MashZone NextGen supports any valid format for the body of a POST request and the appropriate MIME types. Common examples include XML, JSON or plain text.

The URL for a REST web service using POST might look like this:

```
http://www.someOrg.com/myRestService
```

Running Mashables or Mashups and Other Tasks

The most common task when you are working with a mashable or mashup is to "Create a Basic App" on page 1093 following these steps (click any step):

Before you can create an app, you must Run and Preview Mashable/Mashup Data. For new mashables and mashups that you register or create, you may also want to Add Views to Mashables and Mashups so that you and other users can create apps.

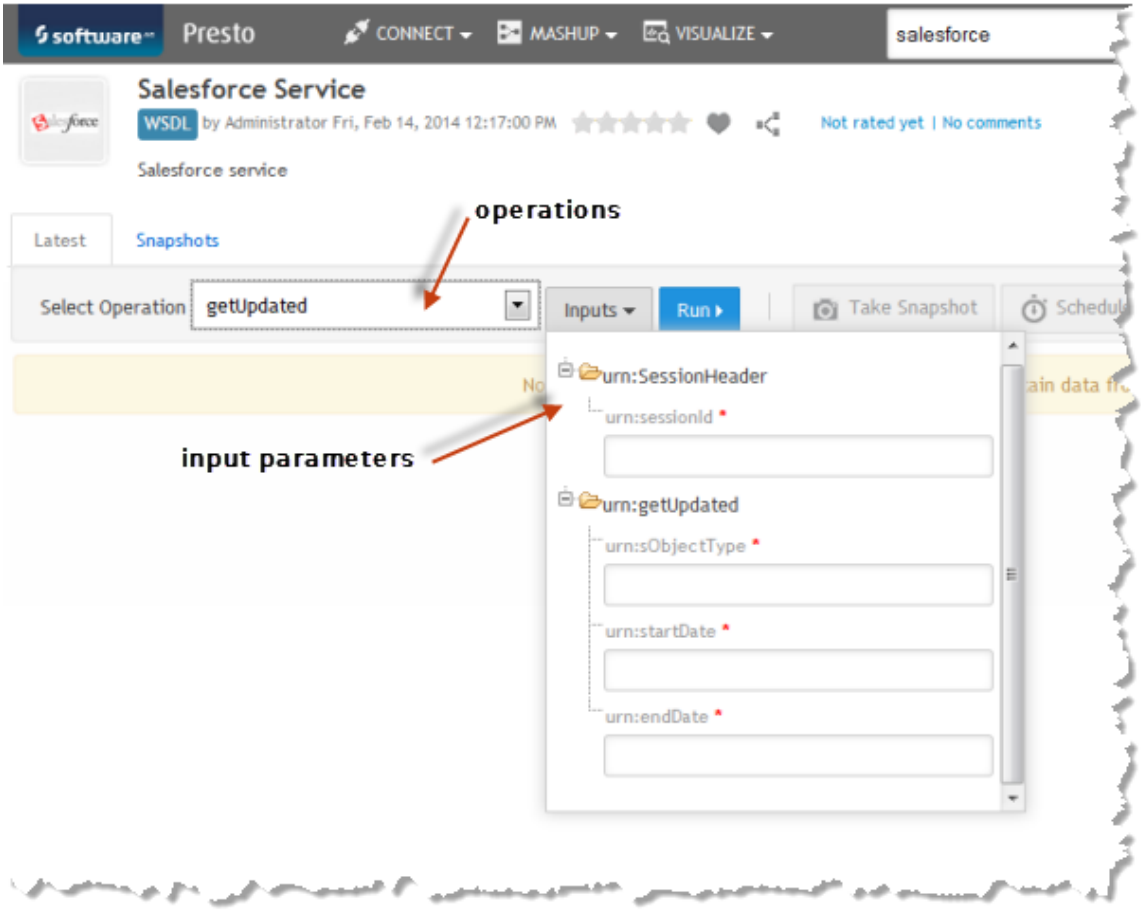
You or a MashZone NextGen administrator must also Grant Permission to Run Mashables, Mashups and Apps to other users. Other common tasks for both mashable information sources and mashups include:

- Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub
- Add to Favorites or Bookmark
- Manage Views
- Take, View or Delete Snapshots
- Schedule Snapshots
- Share With Other Users
- Add Feedback
- Add or Update Meta Data: Description, Category, Provider and Tags
- Add a Thumbnail
- Change the Artifact Name
- Update Mashable Endpoints
- Update Mashable Security Profiles
- Change the Status
- Delete
- Feature or Unfeature
- Use Mashable/Mashup Technical Specs for APIs to invoke these artifacts

Run and Preview Mashable/Mashup Data

You can run a mashup or mashable to play with the mashup or mashable or preview data from the mashup's or mashable's artifact page. You must also run a mashup or mashable *before* you can add views, take a snapshot, schedule snapshots or create a basic app.

1. Open the mashable's or mashup's artifact page from search results, favorites or bookmarks.
2. Open the **Latest** tab, if needed, where you can run and work with current data for this mashable or mashup.
3. If the mashable has several operations, choose the operation you want to work with from the list in the Run toolbar.



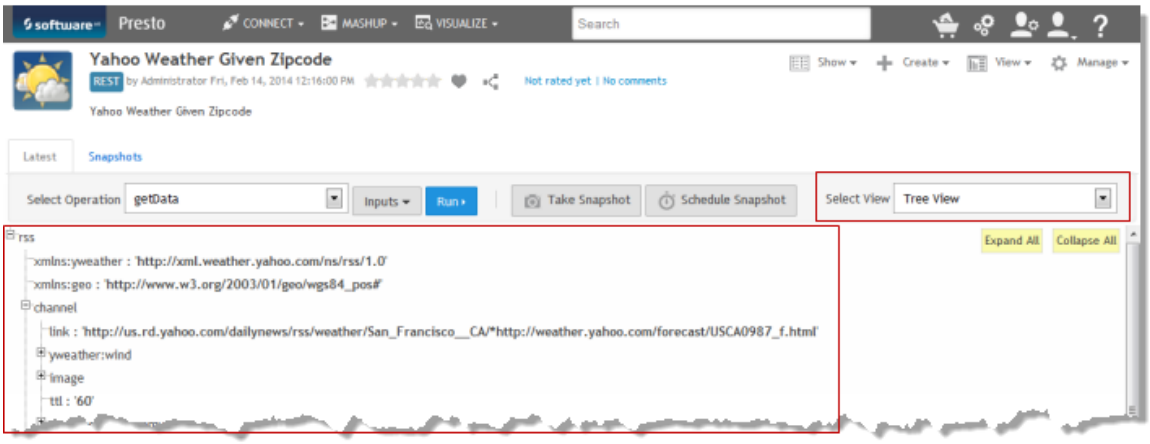
4. Complete the input parameters, if any, for this mashable or mashup.

   The exact inputs you must supply is different for each mashup or mashable. Some may be optional. If you are not sure, try leaving parameters blank and running the mashup or mashup. Or consult with your MashZone NextGen administrator.

   You can enter values directly. See also "Valid Date Formats for MashZone NextGen Mashables and Mashups" on page 179 for more information.

5. Click **Run**.

   Response data from the mashup or mashable displays in the preview area using the default view for this mashup or mashable. If no other views have been added yet, this displays in the Tree View.



You can choose other views that have been configured for the mashup or mashable to see the data in other forms.

Tip: For new mashups or mashables, it is important to add views so that you and other users can work with the mashup or mashable. You must have permissions to add views.

Valid Date Formats for MashZone NextGen Mashables and Mashups

The information source providers for mashables determine what is valid input for the web services or feeds that they provide, including the date and time formats that they support. In most cases in MashZone NextGen, the fields where you enter dates allow you to control the exact format of the date. You must provide dates and times in the expected format for that information source.

WSDL web services, however, commonly use the ISO 8601 date standard and specifies this in the WSDL. If this is set in the WSDL, MashZone NextGen provides a window where you can select a date, and if appropriate a time, and automatically formats your choice to meet this standard.

For mashups and database mashables, however, MashZone NextGen is the information provider. The date and time formats that you can use in input parameters to mashups or database mashables include:

Supported Date Format	Notes/Examples
YYYY/MM/DDThh:mm:ss.sss (ISO 8601)	Time is based on a 24-hour clock (0-23). For example: ■ 1950/06/01 00:01 ■ 2008-10-31 ■ 1999.12.31 23:59:59
YYYY/MM/DD hh:mm:ss.sss	
YYYY-MM-DD hh:mm:ss.sss	

Supported Date Format	Notes/Examples
YYYY.MM.DD hh:mm:ss.sss	
YYYY/MM/DD hh:mm:ss	
YYYY-MM-DD hh:mm:ss	
YYYY.MM.DD hh:mm:ss	
YYYY/MM/DD hh:mm	
YYYY-MM-DD hh:mm	
YYYY.MM.DD hh:mm	
YYYY/MM/DD	
YYYY-MM-DD	
YYYY.MM.DD	

Update Mashable Endpoints

For many mashables, the *endpoint* URL defines both how MashZone NextGen retrieves information from that source and what information is included. Changing the endpoint can have a critical impact on the mashable, and thus have a critical impact on mashups or apps that depend on the mashable.

Changes in the environment in your organization or deployment to new environments, however, may require that mashable endpoints be changed. Care should be taken to ensure that updates to endpoints do *not* change the behavior and results of the mashable.

To support environment changes to mashable endpoints, MashZone NextGen administrators and mashable owners can update endpoints in these cases:

- For *remote* CSV or Excel spreadsheets, you may change any portion of the URL.

  Note: With local CSV or Excel spreadsheets, MashZone NextGen updates a snapshot of the spreadsheet data so there is no endpoint URL.

- For RSS or Atom feeds, you may change any portion of the URL. You *must not* edit or add parameters if the feed accepts parameters.

- For REST web services, you may change any portion of the endpoint URL up to the parameters (before the ? if any). This includes the domain name and path.

  You *cannot* edit or add parameters.

- For WSDL web services, the resource URL points to the WSDL file for the web service. The actual endpoint is identified in the WSDL file.

  You can edit any portion of the endpoint, but you *cannot* edit the URL to WSDL file.

To edit endpoints

1. Find and open the mashable you want to edit.
2. Select **Show >** **Info** and:
   - *For remote CSV or Excel* mashables, click **Edit** next to the Resource URL field. Change the URL in the **New Resource URL** field.
   - *For RSS or Atom* mashables, click **Edit** next to the Resource URL field. Change the URL in the **New Resource URL** field.
   - *For REST* mashables, click **Edit** next to the Resource URL field. You can change the domain and path portion of the URL in the **New Resource URL** field, but not parameters, if any, that come after the `?`.
   - *For WSDL* mashables, click **Edit** next to the Endpoint URL field and update any portion of the endpoint URL to change in this WSDL.
3. Click **Save** and close the Info window.

Update Mashable Security Profiles

Security profiles define the credentials or other security information needed to securely connect to a mashable information source and be authenticated. MashZone NextGen administrators and mashable owners can update security profiles if the protocol or security information changes.

To update a security profile

1. Find the mashable and open its artifact page.
2. Select **Manage >** **Security Profile**.
3. Update the security profile properties as needed.

   See "Mashable Authentication with Security Profiles" on page 161 for information on properties for any of the security profiles bundled in MashZone NextGen. Contact your MashZone NextGen administrator for information on properties for custom security profiles.

   You can also use MashZone NextGen Global or User Attributes in security profile properties. Simply type `$` in a property and choose an available attribute from the

list that opens. See "Using MashZone NextGen Global or User Attributes in Security Profiles" on page 163 for more information.

4. Click **Save these changes**.

Setting a Character Encoding for a Mashable

Character encodings determine what characters are used in mashable results and the actual digital representation of those characters. In most cases, the web services or other information sources underlying mashables send information about the character encoding of the response.

In some cases, this encoding information is missing or incorrect. MashZone NextGen uses a default encoding, but this can cause results to display strange characters not appropriate for the language used with the result or show multiple characters where only one is clearly needed.

You can override this default encoding for individual mashables to fix this problem.

1. Have your MashZone NextGen administrator add or update an artifact attribute for character encodings to use with mashables and ensure that it supports the character encoding being used in responses from this mashable.
   - If the attribute does not yet exist, your MashZone NextGen administrator must add the following attribute:
     - *Mashable* = as the type of artifact that this attribute applies to.
     - **Attribute Name** = `encoding_override`
     - **Datatype** = `ENUM`
     - **Possible Values** = a comma-separated list of character encodings for users to choose from.

       This list *must* include `utf-8` which is the default character encoding used in MashZone NextGen. It can include any character encoding supported by the JDK used with MashZone NextGen. For example:

       ```
       iso-8859-1,utf-8
       ```

       Allows users to choose between the default character encoding and the Latin-1 character encoding that supports most Western European languages.
     - **Default Value** = *must* be left blank.
   - If the attribute exists, your MashZone NextGen administrator should make sure that the character encoding you need is listed as a possible value and that this artifact attribute is available for mashables.
2. Once the attribute exists with the proper information, open this mashable from search results or other links.
3. Select **Show >** **Info**.

4. Find the **encoding_override** field and Click **Edit**.
5. Choose the encoding that this mashable is using and click **Save**.

Take, View or Delete Snapshots

Each time you run a mashable or mashup in its artifact page, you can choose to save these results as a *snapshot*. See "Take a Snapshot" on page 183 for instructions.

> **Note:** The snapshot feature is only available if your MashZone NextGen administrator has enabled this feature.

From mashable or mashup artifact pages you can also Schedule Snapshots, "Register a Snapshot as a Mashable" on page 158 or "Delete a Snapshot" on page 185.

Take a Snapshot

1. Open a mashable or mashup and Run and Preview Mashable/Mashup Data.
2. Choose the view you want to use to see snapshot data. If needed, add a new view (see "Add Views to Mashables and Mashups" on page 901 for instructions).
3. Click **Take a Snapshot** and complete these fields:
   - **Snapshot Name** = this defaults to `Snapshot` plus the current date and time. Change this if needed.

     > **Tip:** It is a good practice to keep the date and time to help differentiate this snapshot from other snapshots taken with the same input at different times.

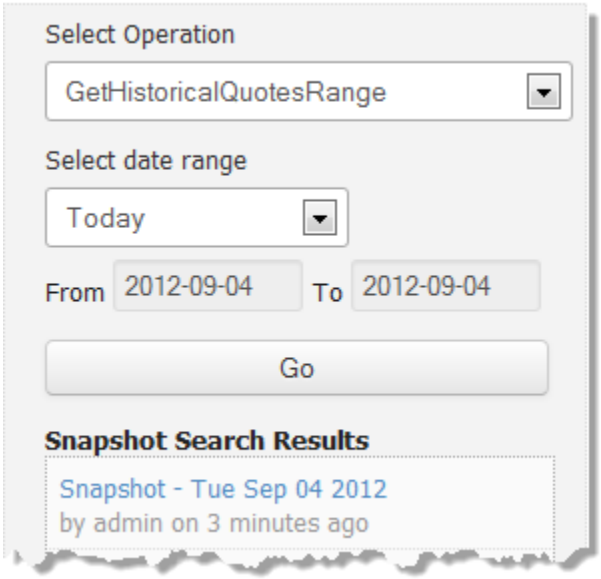
   - **Description** = an optional short description.

     > **Tip:** It is a good practice to provide a description to clarify the parameters, if any, used for this snapshot or to provide other identifying information, especially if you use the default snapshot name.

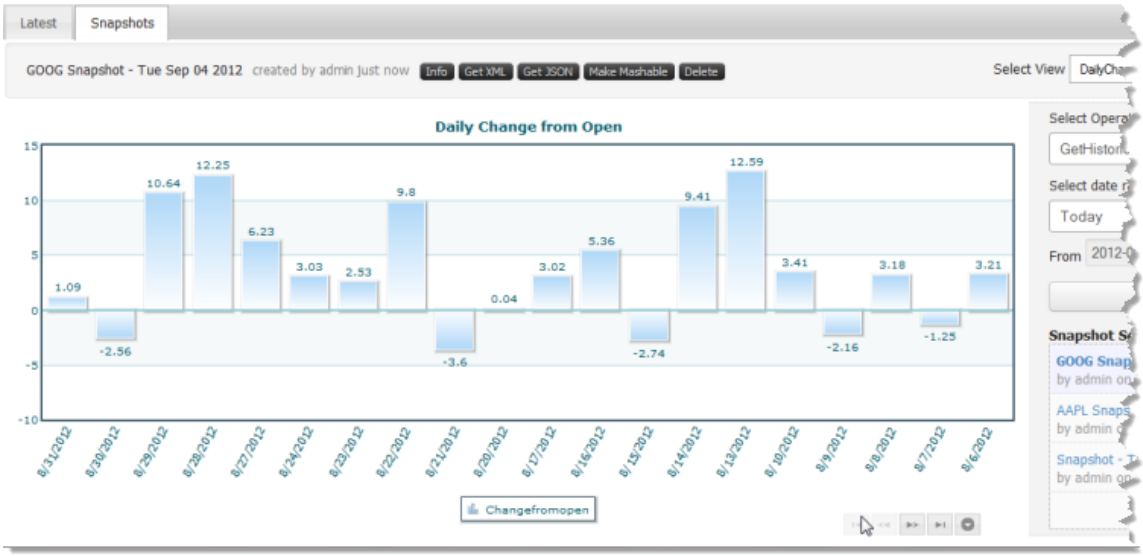
4. Click **Ok**.

Find and View Snapshots

1. Open a mashable or mashup from search results, links or favorites.
2. Click the **Snapshots** tab, if needed. This shows a list of snapshots taken today for this mashable operation or mashup.



3. If needed, select a different operation or select or enter a date range and click **Go** to find the specific snapshot you want to see.
   - **Select Operation** = for mashables with several operations, choose the operation that was used to take the snapshot you want to see.
   - **Select date range** = to see snapshots taken today, the past 7 days, past month and so on.
   - **From** and **To** = to select a range of specific dates. Enter dates in YYYY-MM-DD format.
4. Click a snapshot summary to view these results.



View Snapshot Results in XML or JSON

You can view snapshot data in XML or JSON (JavaScript Object Notation) formats. Simply find the snapshot, view it and click **Get XML** or **Get JSON** to view the data in another browser tab or window.

Delete a Snapshot

Carefully consider before you delete a snapshot. If mashups or apps use this snapshot, deleting it will cause the mashup or app to fail.

1. Find the snapshot you want to delete and view snapshot data (see "Find and View Snapshots" on page 183 for instructions).
2. Click **Delete** and confirm this when prompted.

Schedule Snapshots

You can have MashZone NextGen automatically run mashables or mashups and take snapshots based on a schedule that you define. See "Schedule Snapshot Jobs" on page 185 for instructions. You can also Delete Snapshot Scheduled Jobs.

Schedule Snapshot Jobs

1. Open a mashable or mashup and Run and Preview Mashable/Mashup Data using the input parameters, if any, that you want to use for each snapshot in this schedule.
2. Choose the view that you want to use to see data for these snapshots.

   If needed, add a new view (see "Add Views to Mashables and Mashups" on page 901 for instructions).
3. Click **Schedule Snapshot**.
4. Choose the frequency that snapshots should be taken in the **Schedule** field.

   Note: Jobs that are scheduled in second, minute or hourly intervals always have a defined termination point. Jobs scheduled with a daily, weekly or monthly interval have *no* termination date. You must delete these scheduled jobs to stop them.
5. For `Seconds`, `Minutes` or `Hourly` schedules, complete these properties:
   - **Run Every** = the time interval between scheduled snapshots.
   - **How Many Times** = the total number of snapshots to take.
6. For `Daily` schedules, define the time of day to take these snapshots in the **At What Time** property.
7. For `Weekly` schedules, choose the **Day of Week** and **At What Time** to take the snapshots.

8. For `Monthly` schedules, choose the **Day of Month** and **At What Time** to take the snapshots.
9. Click **Schedule Snapshot**.

   This creates the scheduled job with a name using the mashable or mashup name, the name of the operation and the date and time you created the schedule. Click **Close**.

Delete Snapshot Scheduled Jobs

1. Open a mashable or mashup and Run and Preview Mashable/Mashup Data.
2. Click **Schedule Snapshot**.
3. A list of snapshot schedules displays at the bottom of the Schedule A Snapshot window. Click **Delete**.

4 Dashboards

- Use dashboards in view mode 188
- Manage dashboards in edit mode 192
- Configure dashboard views 199
- Configure dashboard components 200
- Customize desktop 279

The MashZone NextGenDashboard component gives you an easy, graphic way to create, manage and view your dashboards.

Dashboards are interactive applications that collects data from different data sources, combine it, and visualize it. The data can come from different sources and is combined. Possible data sources include *Event data source* or *BigMemory data source*. Dashboards are composed of individual components (e.g.,Real-time Charts, MashZone NextGen App, MashZone NextGen Mashup and MashZone NextGen Grid), which obtain their data from data sources and display it. You can combine the individual components to filter the displayed results interactively and thus analyze them intuitively.

Use dashboards in view mode

In the view mode of the MashZone NextGenDashboard component you can display and use your dashboards interactively.

Display dashboards in MashZone NextGen

You can display your dashboards with actual data in the MashZone NextGenDashboard component.

Procedure

1. Click **Dashboard** > **Open** in the MashZone NextGen Hub main menu.
2. Select an **Available Dashboard** and click **OK**. The selected dashboard will be displayed in MashZone NextGen Dashboard component.

   You can also search a dashboard by keyboard entry.

The dashboard will be displayed with actual data in the view mode.

Click the **Edit dashboard** icon in the Dashboard main menu to switch from view mode to the edit mode.

Click the **View dashboard** icon in the Dashboard main menu to switch from edit mode to the view mode.

Use interactive filter in your dashboards

You can use interactive filters for charts and tables, if the relevant filter conditions have been defined for these components. You can set a filter by clicking a data element of a component, e. g., a data point in a line chart. The selected data element of this component takes effect as a filter for an associated component.

Procedure

1. "Display dashboards in MashZone NextGen" on page 188

2. Click on a data element of a dashboard component, e. g., a coordinate of a line chart or a cell in a table.

   The selected data element is applied as a filter to all associated components.

3. Click the **Menu** icon > **Delete Selection** to remove the set filter.

The dashboard components will be displayed with the selected filters.

You can define filters for specific dashboard components. See "Define filters for dashboard components " on page 276.

Use dynamic URL selection

By specifying URL parameters you can dynamically select specific elements of dashboard components, e.g. a grid row, a combobox entry, a pie slice, etc. When you call a dashboard via URL in the view mode the elements are automatically preselected and, if applicable, the preselected elements are used as filter or action trigger.

You can select the URL parameters required for dynamic URL selection in a dashboard component's properties menu and copy them to the clipboard. You can set a preselection based on data columns for each dashboard component that supports data preselection, e. g. Grid, Input field or several charts.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component that supports data preselection. The relevant properties dialog is displayed.
3. Before you set a preselection you have to assign the relevant data columns to the component. See Assign data sources to dashboard components.
4. Click the **URL selection** tab.
5. You can enter a component id in the **URL-ID** input field. If you change the preset id you have to choose a unique id within your dashboard and you have to save the dashboard to persist the new **URL-ID**.
6. The **URL example** field contains an URL selection of the components including all assigned data columns and corresponding dummy values.
7. You can edit the URL selection according to your requirements in the **URL example** field.

   Expected formats:

   - number : no thousands separator and a dot for decimal separator
   - date / time : yyyy-MM-ddThh:mm:ss
   - text : no restrictions (just URL encoded)

8. Copy the URL selection to the clipboard. If your browser supports this function a **Copy to clipboard** button is available.

   The URL parameters get copied to the clipboard of your operating system (URL encoded).

9. Add the parameters to an URL that you want to use for calling a dashboard and enter the URL in your web browser.

The dashboard will be displayed with the preselection in the view mode.

Example

Preselection parameters

&cn16.Time=2015-12-23&cn16.Location=New%20York

URL with added parameters

http://<local host>:8080/presto/hub/dashboard/dashboard.jsp?
editmode=false&guid=0bd1cbcc-49d2-4cb1-a5fe-72cfdc624cda
&cn16.Time=2015-12-23&cn16.Location=New%20York

These URL parameters get applied when you open the dashboard. To apply a modified preselection, you need to reload the dashboard page. But you can also apply a selection in an opened dashboard without reloading the whole page. For that, the selection string has to start with #... instead of &...

...**#**cn16.Time=2015-12-23&cn16.Location=New%20York

Display dashboards without application header

In the view mode you can display your dashboards without the application header of the MashZone NextGen. Only the dashboard will be displayed in the web browser tab.

Procedure

1. Open a Dashboard in the dashboard view mode.
2. You can hide the application header by adding the `appheader=false` parameter to the dashboard URL in the view mode in the following form.

   http://<url to dashboard>?appheader=false

3. In addition, you can display a drop-down menu in the dashboard which provides zoom, help and logout options. The menu icon will be displayed in the top right corner. Add the `showmenu=true` parameter to the dashboard URL in the following form.

   http://<url to dashboard>?appheader=false&showmenu=true

The dashboard will be displayed without the application header in the view mode.

URL example

http://<local host>:8080/presto/hub/dashboard/dashboard.jsp?editmode=false&guid=0bd1cbcc-49d2-4cb1-a5fe-72cfdc624cda&appheader=false&showmenu=true

Refresh data of dashboard components

You can refresh manually the data evaluated in a dashboard component currently displayed.

The usage of the manual **Refresh** option is independent of the Auto Refresh.

The data is extracted from the cache or recalculated if the Refresh Rate of the data source has expired. The default Refresh Rate value is 30 sec. You can set the **Refresh Rate** in the Assign data dialog.

Procedure

1. "Display dashboards in MashZone NextGen" on page 188
2. Click on a dashboard component.
3. Click the **Menu** icon > **Refresh**.

The dashboard component will be displayed with the refreshed data.

Most of the dashboard components provide the **Auto Refresh** option. With this option you can activate the automatic data retrieval for a component. According to the set Refresh Rate the source data are read in again and recalculated automatically. The **Auto Refresh** option is available on the **Config** tab in the component properties. In the dashboard edit mode click on an inserted component to display the properties dialog.

Change column width and column sort order

In the dashboard view mode you can change the initial table column width and initial table column sort order of a **Grid** component.

Procedure

1. Display a dashboard in the dashboard view mode.
2. Click on a table column header of an inserted **Grid** component.

   The first click changes the sort order to ascending, the second click to descending and the third click to unsorted, if this was the initial state.

   You can sort several columns at the same time. To select more than one column press the **Shift** key and click on the required column headings. In this case the rows are sorted by the first column.
3. To adapt the column width you can drag the column borders with the mouse.

The table is displayed according to your configurations.

Manage dashboards in edit mode

In the edit mode of the MashZone NextGenDashboard component you can manage your dashboards.

Create a dashboard

You can create dashboards in the MashZone NextGenDashboard component.

Procedure

1. Click **Dashboard** > **Create** in the MashZone NextGen Hub main menu. An empty dashboard will be displayed.
2. If you have already opened a dashboard in MashZone NextGen Dashboard click **Manage** > **New Dashboard** in the Dashboard main menu. An empty dashboard will be displayed.
3. Enter your changes.
4. Click **Manage** > **Save as** in the MashZone NextGen Dashboard main menu. The **Dashboard Properties** dialog will be displayed.
5. Enter the mandatory **Name** of the dashboard.
6. You can enter an optional **Description**.
7. You can optionally enter comma separated search tags in the **Tags** field.
8. Click **OK**.

Your changes are applied.

You can also "Edit dashboard properties" on page 193 later.

To display the dashboard in the view mode click on the **View dashboard** icon in the MashZone NextGen Hub main menu.

Edit a dashboard

You can edit existing dashboards in MashZone NextGenDashboard.

Procedure

1. Click **Dashboard** > **Open** in the MashZone NextGen Hub main menu.
2. Select an **Available Dashboard** and click **OK**. The selected dashboard will be displayed in MashZone NextGen Dashboard component.

In case, there are some local changes for the selected dashboard, a warning message will be displayed. You can click **Continue with Unsaved Local Version** to open the dashboard with the local changes.

3. Enter your changes.
4. Click **Manage** > **Save** in the Dashboard main menu.

Your changes are applied.

To display the dashboard in the view mode click on the **View dashboard** icon in the MashZone NextGen Hub main menu.

Edit dashboard properties

You can edit the properties (name, description and tags) of existing dashboards in MashZone NextGenDashboard.

Procedure

1. Click **Dashboard** > **Open** in the MashZone NextGen Hub main menu.
2. Click **Manage** > **Open** in the Dashboard main menu.
3. Select an **Available Dashboard** and click **OK**. The selected dashboard will be displayed in MashZone NextGen Dashboard component.
4. Click **Manage** > **Properties** in the Dashboard main menu. The **Dashboard Properties** dialog will be displayed.
5. Enter the mandatory **Name** of the dashboard.
6. Enter an optional **Description**.
7. You can optionaly enter comma separated search tags in the **Tags** field.
8. Click **OK**.
9. Click **Manage** > **Save** in the Dashboard main menu.

Your changes are applied.

Delete a dashboard

You can delete dashboards in MashZone NextGenDashboard.

Note: Deleted dashboards couldn't be restored.

Procedure

1. Click **Dashboard** > **Open** in the MashZone NextGen Hub main menu.
2. Select an **Available Dashboard** and click **OK**. The selected dashboard will be displayed in MashZone NextGen Dashboard component.

3. Click **Manage** > **Delete** in the Dashboard main menu.
4. Click **Yes**.

The selected dashboard is deleted.

Manage dashboard permissions

You can manage the permissions of dashboards in MashZone NextGenDashboard. You can assign particular access rights to individual users or withdraw them again. For user groups, you assign access rights to or withdraw them from all members of a group at the same time.

Note: You can assign access rights only for saved dashboards.

Access right

- **Edit**

  This user can display and edit dashboards in MashZone NextGenDashboard.

- **View**

  This user can view dashboards in MashZone NextGenDashboard.

  Procedure

1. Click **Dashboard** > **Open** in the MashZone NextGen Hub main menu.
2. Select an **Available Dashboard** and click **OK**. The selected dashboard will be displayed in MashZone NextGen Dashboard component.
3. Click **Manage** > **Permissions** in the Dashboard main menu. The **Manage Dashboard Permissions** dialog will be displayed.
4. Enter a therm in the search field an click **Search**. Clicking on **Search** without any input values fetches all the users and groups.
5. Click **ShowMashZone NextGendefault groups** to show only default MashZone NextGen users or user group in the **Search results** field..
6. Drag an user or user group from the **Search Result** field and drop it into the **Principals with Permissions** field.
7. Activate or deactivate the **View** or **Edit** privileges of an user or user group.
8. Click **Save**.

Your changes are applied.

If you want to remove a user or user group from the **Principals with Permissions** list then click the **Delete** icon.

Change dashboard style template

You can assign a different style template to an available dashboard. By means of style templates you can customize the look and feel of your dashboards, e. g., colors schemes, fonts or background color.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click **Manage** > **Change style template** in the Dashboard main menu.
3. Select a style template in the drop-down template.
4. Click **OK**.

The selected style template applies to the current dashboard.

You can create your own dashboard style templates, see "Create dashboard style templates" on page 195.

Create dashboard style templates

You have MashZone NextGen administrator permissions.

You can create your own Dashboard dashboard style templates. Creating your own style templates enables you to customize the look and feel of Dashboard dashboards, e. g., colors schemes, fonts or background colors. You can copy and edit the default style template that is supplied with MashZone NextGenDashboard

The dashboard style template files are located in the following folder on the MashZone NextGen server. The default dashboard template file is named default.less.

<*MashZone NextGen installation*> \apache-tomee-jaxrs\webapps\presto\hub\dashboard \assets\custom-look-and-feel\dashboard

Procedure

1. Create a copy of the default style template file and give the file a name of your choice.
2. Open the new template file in your text editor and edit the style parameters according your requirements.
3. Save your changes.
4. Load the new style template file into MashZone NextGen Dashboard.
   a. Open MashZone NextGen Dashboard
   b. Click **Manage** > **Change style template** in the Dashboard main menu.

c. Select the new dashboard style template in the drop-down menu and click **Compile selected to CSS**.

5. Click **OK**.

The new style template is now available in MashZone NextGenDashboard.

You can assign the new style template to your dashboards, see "Change dashboard style template" on page 195.

Edit style templates

You have MashZone NextGen administrator permissions.

You can edit the style templates supplied with MashZone NextGenDashboard. Editing the style templates enables you to customize the look and feel of your dashboards and the Dashboard component itself, e. g., colors schemes, fonts, brand logo or background colors.

The application style template file application.less is located in the following folder on the MashZone NextGen server.

<*MashZone NextGen installation*> \apache-tomee-jaxrs\webapps\presto\hub\dashboard \assets\custom-look-and-feel\application.

The dashboard style template files are located in the following folder on the MashZone NextGen server. The default dashboard template file is named default.less.

<*MashZone NextGen installation*> \apache-tomee-jaxrs\webapps\presto\hub\dashboard \assets\custom-look-and-feel\dashboard

The style templates provide basic variables that are used as default by many components, e.g. @header-color. You can override the basic variables by defining more specific, customized variables, e.g. @header-dashboard-name-color.

For example, if @header-color is black and @header-dashboard-name-color is red and both variables are the only ones defined, all labels in the header will be black except of the dashboard name label, it will be red.

All available application and dashboard variables are described in the corresponding style template files.

Procedure

1. Open the relevant template style file in your text editor.
2. Edit the style parameters according your requirements.
3. Save your changes.
4. Reload the changed style template files into MashZone NextGen Dashboard.
   a. Open MashZone NextGen Dashboard
   b. Click **Manage** > **Change style template** in the Dashboard main menu.

c. To reload the application style template click **Compile to CSS**.

d. To reload the dashboard style template select the relevant dashboard style template in the drop-down menu and click **Compile selected to CSS**.

5. Click **OK**.

The changed style template has been reloaded into MashZone NextGenDashboard and the style templates apply to the application or dashboard.

Importing Dashboards

You can import dashboards to MashZone NextGenDashboard.

The dashboards are saved in a zip file that contains among other things the dashboard definition, resource policy and dashboard permissions. If you import a dashboard including the permissions then the creator of the dashboard can view and edit the dashboard. Importing dashboards without the relevant permissions makes the importer automatically to the creator of these dashboards.

Procedure

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter this command:

```
padmin importDashboards [-l prestoURL] -f input-file -p importPermissions
-u username -w password [-v] [-o]
```

- *-f input-file* : path and name for the import zip file.
- -p *importPermissions* Imports the resource policy and permissions saved in the import zip file.

  If you import dashboards without permissions makes the importer automatically to the creator of these dashboards and the dashboards has no explicit permissions which means that only administrators can see and work with the dashboards.
- *-o*: is optional. Allows to overwrite an existing dashboard in MashZone NextGenDashboard.
- *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/esd/api`.
- *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
- *-w password* : is the MashZone NextGen password to log in with.
- *-v*: is an optional flag to turn on verbose logging.

Once the import command completes successfully, you can use the imported dashboards in MashZone NextGenDashboard.

If you have imported dashboards from MashZone NextGen 3.9 in MashZone NextGen 9.9 then save the imported Dashboards in the edit mode of MashZone NextGenDashboard before you display them in the view mode. Otherwise an error message will be displayed.

Exporting Dashboards

You can export your Dashboard dashboards.

Export creates an export file that you can then use to import dashboards to MashZone NextGenDashboard.

Procedure

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter this command:

```
padmin exportDashboards -i identifier [-f output-file] [-l prestoURL]
-u username -w password [-v] [-o]
```

- -i *identifier* : mandatory dashboard identifier. It can be "id=", "name=" or "all", enclosed in quots.

  -i "name=dashboardname": If there are multiple dashboards with the same name then only the first founded dashboard will be exported.

  -i "id=43243244434432": The dashboard id (Guid) is unique in the MashZone NextGen system.

  -i "all": Export of all dashboards for that user.

- *-f output-file* : an optional path and name for the export file to put dashboards. If omitted, this generates an output zip file in the folder where this command is executed:
  - Single export with option -i "id=3456" or "name=name" create a new file with name "name_guid.zip"
  - Multiple export with option -i "all" create a new file dashboard-export-timestamp.zip

  This file must not already exist, unless you also use the `-o` option.

- *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/esd/api`.
- *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
- *-w password* : is the MashZone NextGen password to log in with.
- *-v*: is an optional flag to turn on verbose logging.

- *-o*: an optional flag to overwrite an existing export file. If you omit this option, the output file must not already exist.

Once the export command completes successfully, you can use the output file to import dashboards to MashZone NextGenDashboard.

Permissions for each Dashboard were automatically stored into the zip file. If there are not any permissions assigned to the dashboard an empty permission file is stored.

There is also an information about the dashboard creator stored in the zip file.

Configure dashboard views

You can configure the view of your dashboards in the MashZone NextGenDashboard component.

Add views in a dashboard

You can add additional views in a dashboard. A dashboard can consist of several views. The individual views are displayed independently on separate tabs.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click on the **New tab** icon beside the tab name.

In addition a new view is created and displayed as a tab.

Delete views in a dashboard

You can delete any view from a dashboard. A dashboard view is displayed as a separate tab.

Note: Deleted dashboard views couldn't be restored.

Procedure

1. "Open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Move the mouse over the title of the relevant tab.
3. Click the **Delete** icon in the pop-up menu.
4. Click **Delete**.

The selected dashboard view is deleted.

Configure dashboard views

You can configure a dashboard view. Specify a name for the dashboard view and set the background color.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Move the mouse over the title of the relevant tab.
3. Enter a text in the **Name** box in the pop-up menu.
4. Set the Background color for the view.
   a. Click the color selection field to set the background color of the view.
   b. Select a color.

Your dashboard view is configured.

Configure dashboard components

You can configure the components of your dashboards in the MashZone NextGenDashboard component.

Insert components in a dashboard

You can use various dashboard components to insert items such as Line Chart, Speedometer Chart, Mashups, Apps or Tables into your dashboard.

Available components:

- Line Chart
- Column Chart
- Bar Chart
- Pie Chart
- Grid
- Speedometer Chart
- Traffic Light
- Drop-down box
- Input field
- MashZone NextGen App

- MashZone NextGen Views
- Image
- Label

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click a component icon or place the icon via drag and drop. The selected component is inserted as simple, graphic blocks in the dashboard.
3. Click the inserted graphic block and place the graphic on the dashboard per drag and drop.

The selected component is inserted and placed in the dashboard.

Insert Line Charts in a dashboard

You can insert a Line Chart in your dashboard.

A Line Chart can display values for two iterations. Two dimensions and one KPI can be used, or one dimension and multiple KPIs. The second iteration is displayed with several stacked lines. Multiple KPIs are then displayed using lines of different colors.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Line Chart** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properies dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Line chart" on page 213 for details.
4. Click the **Configure filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** Tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Enter an optional **Chart Title** for the displayed chart of the component.
9. Select a **Line Type**. Available are **linear**, **curved** or **step** line types.
10. Activate the **Interpolate** option.

    Activated option: Missing values are ignored (leads to a continuous line).

    Deactivated option: Missing values lead to gaps within the line.

11. Activate the **Legend** option to display a legend in the chart component.
12. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.
13. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

A Line Chart will be displayed with real data of the assigned data source.

Insert Column Charts in a dashboard

You can insert a Column Chart in your Dashboard dashboard.

A Column Chart can display values for two iterations. Two dimensions and one KPI can be used, or one dimension and multiple KPIs. The second iteration is displayed with several stacked columns. Multiple KPIs are then displayed using columns of different colors.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Column Chart** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Click the **Assign data** icon to edit the data source assignment. See "Assign data sources to Column chart" on page 215 for details.
4. Click the **Configure filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Enter an optional **Chart Title** for the displayed chart of the component.
9. Select a **Column Type**. This option is only available if a data source has been assigned. Available are **Stacked** or **Grouped** columns.
10. Activate the **Legend** option to display a legend in the chart component.
11. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.
12. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

A Column Chart will be displayed with real data of the assigned data source.

Insert Bar Charts in a dashboard

You can insert a Bar Chart in your Dashboard dashboard.

A Bar Chart can display values for two iterations. Two dimensions and one KPI can be used, or one dimension and multiple KPIs. The second iteration is displayed with several stacked columns. Multiple KPIs are then displayed using bars of different colors.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Bar Chart** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Bar chart" on page 216 for details.
4. Click the **Filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Enter an optional **Chart Title** for the displayed chart of the component.
9. Select a **Bar Type**. This option is only available if a data source has been assigned. Available are **Stacked** or **Grouped** columns.
10. Activate the **Legend** option to display a legend in the chart component.
11. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.
12. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

A Bar Chart will be displayed with real data of the assigned data source.

Insert Pie Charts in a dashboard

You can insert a Pie Chart in your Dashboard dashboard.

A Pie Chart can display one numeric KPI iterated via a dimension (text or date dimension).

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.

2. Click the **Pie Chart** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Pie chart" on page 218 for details.
4. Click the **Configure filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Enter an optional **Chart Title** for the displayed chart of the component.
9. Select a **Inner Radius** to display a inner circle with the selected radius.
10. Select in the **Values** drop-down menu the kind of display for the displayed KPIs.
11. Select in the **Values Position** drop-down menu the position where the KPI values are displayed. Available positions are **Inside** and **Outside** of the Pie Chart.
12. Activate the **Legend** option to display a legend in the chart component.
13. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.
14. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

A Pie Chart will be displayed with real data of the assigned data source.

Insert Table in a dashboard

You can use the **Grid** component to insert a table in your Dashboard dashboard.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Grid** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Grid" on page 220 for details.
4. Click the **Filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Edit the display options of the Grid component.
6. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.

7. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

A table will be displayed with real data of the assigned data source.

Edit Table display options

You can edit the **Grid** component display options.

Procedure

1. Click on an inserted **Grid** component in a dashboard. The properties dialog is displayed.
2. Click the **Config** tab.
3. Enter an optional component **Name** on the **Config** tab.
4. Set the **Container** header and border.
   - Click the **Hide Header** icon. Hides the header incl. the title of the component and the content will be resized. Click the icon again to display the header.
   - Click the **Hide Border** icon. Hides the outline of the component container. Click the icon again to display the outline.
5. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.

   See "Refresh data of dashboard components" on page 191 for details.
6. Click the **Configuration** icon to edit the Grid component interactively. In this edit mode the Grid component is displayed with a blue frame. Now you can set the initial widths and the initial sort order of the columns. You can change the sort order by clicking on the column header. To adapt the column width you can drag the column borders with the mouse. If desired, you can change the defined column width and column sort order in the dashboard view mode.
7. Activate the **Auto column with** option to adapt the column with to the column content automatically. The horizontal scroll bar is no longer displayed in the component.
8. Click the **Style** tab.
9. Select a **Column** in the corresponding drop-down menu. For the selected column you can set own styles.
10. Select a style for the column header in the **Header style** drop-down menu.
11. Select a style for the column cells in the **Style** drop-down menu.
12. Select the **Alignment** of the column values. Default is left aligned.

The table will be displayed according to your configurations.

Insert Speedometer Charts in a dashboard

You can insert a Speedometer Chart in your Dashboard dashboard.

A speedometer chart displays a set of aggregated KPI values. The value ranges are arranged in a semicircle with a red pointer that indicates the actual value of the KPI.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Speedometer Chart** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Speedometer chart" on page 224 for details.
4. Click the **Filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name**.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Activate the **Scale** option to display a scale in the chart.
9. Activate the **Scale Value** option to display the scale values in the chart.
10. Activate the **KPI Name** option to display the KPI name in the chart.
11. Activate the **KPI Value** option to display the KPI value in the chart.
12. Activate the **Threshold** option to display the threshold in the chart.
13. Activate the **Threshold Value** option to display the threshold value in the chart.
14. Activate the **Level Meter** option to display a level meter in the chart.
15. Activate the **Percentage** option to display a scale from 0 to 100% n the chart. The KPI value is also displayed in percent.

A Speedometer Chart will be displayed with real data of the assigned data source.

Insert Horizontal and Vertical Gauge Charts in a dashboard

You can insert a Horizontal and Vertical Gauge Chart in your dashboard.

A Horizontal or Vertical Gauge chart displays a set of aggregated KPI values. The value ranges are arranged in a horizontal or vertical bar with a pointer that indicates the actual value of the KPI.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Move the mouse pointer over the **Speedometer Chart** component icon.

3. Click the **Horizontal Gauge Chart** or the **Vertical Gauge Chart**component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
4. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Horizontal and Vertical Gauge Chart" on page 226 for details.
5. Click the **Filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
6. Enter an optional component **Name**.
7. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
8. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
9. Activate the **Scale** option to display a scale in the chart.
10. Activate the **Scale Value** option to display the scale values in the chart.
11. Activate the **KPI Name** option to display the KPI name in the chart.
12. Activate the **KPI Value** option to display the KPI value in the chart.
13. Activate the **Threshold** option to display the threshold in the chart.
14. Activate the **Threshold Value** option to display the threshold value in the chart.
15. Activate the **Level Meter** option to display a level meter in the chart.
16. Activate the **Percentage** option to display a scale from 0 to 100% in the chart. The KPI value is also displayed in percent.
17. Select the scale size in the **Scale Size** drop down menu bar.

A Horizontal or Vertical Gauge Chart will be displayed with real data of the assigned data source.

Insert Traffic Lights in a dashboard

You can insert a Traffic Light in your dashboard.

A multi-color vertical, horizontal or single Traffic Light shows the threshold range in which a KPI value is located. You can define thresholds in the **Assign data** dialogue.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Traffic Light** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Traffic Light" on page 228 for details.

4. Click the **Configure filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Select a traffic light **Type**. Available are **Vertical**, **Horizontal** and **Single**.
9. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.

A Traffic Light will be displayed with real data of the assigned data source.

Insert Drop-down box in a dashboard

You can insert a Drop-down box in your Dashboard dashboard.

The Drop-down box provides you with a selection of values in a drop down menu for filtering other dashboard components.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Drop-down box** component icon. The selected component is inserted in the dashboard and the properties dialog is displayed.
3. Click the **Configure** icon to edit the data source assignment. See "Assign data sources to Drop-down box" on page 231 for details.
4. Click the **Configure filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Activate the **No selction** option.

   If this option is activated, a **No selection** value will be added to the value list as first entry. Initially the first entry of the value list is preselected in the Drop-down box for filtering. Selecting the **No selection** value has no effect on filtering other components.
9. Enter an alternative text in the **Entry** field.

   The text will be displayed in the Drop-down box instead of the default text **No selection**.

10. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.
11. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

A Drop-down box will be displayed with real data of the assigned data source.

Insert Input field in a dashboard

You can insert a Input field in your Dashboard dashboard.

The Input field enables you to enter values manually for filtering other dashboard components. See for details.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Input field** component icon. The selected component is inserted in the dashboard and the properties dialog is displayed.
3. Enter an optional component **Name** on the **Config** tab.
4. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
5. Select a **Data type** for the user input. The user can only insert values of the selected data type. Click on the **Text**, **Number** or **Date** icon.
6. Select a **Format** in the drop-down menu. The user can only insert values with the selected format.

   The option is only available for **Date** data type.
7. Insert an optional **Initial value**. The **Initial value** will be displayed in the **Input field** as default preset value.
8. Activate the **Current date/time** option to insert the current date in the **Initial value** box.

   The component always uses the current date and time of the dashboard at runtime. The option is only available for **Date** data type. The value displayed depends on the format selected.
9. Enter an alternative text in the **Submit button text** box if necessary .

   The text will be displayed in the **Input field** instead of the default text **Ok**.
10. Activate the **Show submit button** option to display the submit button in the **Input field**.
11. Activate the **Submit value with each keystroke** option if necessary. The **Input field** will submit the entered values with each keystroke immediately.

    For **Date** and **Number** the **Input field** will only submit valid date or number values .

12. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.
13. Specify **URL selections** for the component, if required. See "Use dynamic URL selection" on page 189 for details.

An **Input field** will be displayed with the preset value.

Insert MashZone NextGen Apps in a dashboard

You can use the **MashZone NextGen** component to insert a MashZone NextGen app in your Dashboard dashboard.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **MashZone NextGen** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Enter an optional component **Name**.
4. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
5. Click the **Configure** icon to edit the data source assignment. See "Assign data sources to MashZone NextGen App" on page 230 for details.
6. Click the **Refresh** icon to reload the data of assigned data sources.

A MashZone NextGen App will be displayed with real data of the assigned data source.

Insert MashZone NextGen Mashups in a dashboard

You can use the **MashZone NextGen Views** component to insert a MashZone NextGen Mashups in your Dashboard dashboard.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **MashZone NextGen Views** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Enter an optional component **Name**.
4. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
5. Click the **Configure** icon to edit the data source assignment. See "Assign data sources to MashZone NextGen Views" on page 230 for details.
6. Click the **Refresh** icon to reload the data of assigned data sources.

A MashZone NextGen Mashup will be displayed with real data of the assigned data source.

Insert Image in a dashboard

You can insert an Image in your Dashboard dashboard.

The image component supports the file format PNG, GIF and JPEG. The image file can be picked up from a web source or a local folder.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Image** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Enter an optional component **Name**.
4. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
5. Enter a **Source URL** to the image source. The URL can refer to an internet source (http) or a locally hosted resource uploaded to the MashZone NextGen repository. The local URL has the following form: /presto/files/<*resource name*>

   You can upload your images to the MashZone NextGen repository, see "Add External Resources as MashZone NextGen Files" on page 1746 for details.
6. Activate the **Original** option to display the image in the original size.
7. Activate the **Scaled** option to display the image fitted to the component frame size.
8. Activate the **Aspect ratio** option to keep the aspect ratio of the image.
9. Select the **Horizontal alignment** in the drop-down menu to align the image within the component frame.
10. Select the **Vertical alignment** in the drop-down menu to align the image within the component frame.
11. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.

An image will be displayed in the dashboard.

Insert Label in a dashboard

You can insert a Label in your Dashboard dashboard.

A Label displays a fixed text you have entered, or a text that is supplied dynamically by a datasource.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the **Label** component icon. The selected component is inserted as simple, graphic block in the dashboard and the properties dialog is displayed.
3. Enter text in the text field.

   You can enter a fixed text manually or assign a dynamic text from a data source.

   - Double-click the text field and enter a text manually. To enter several lines increase the size of the label component by pulling the anchor points with your mouse and enter line breaks.
   - Click the **Assign Data** icon to edit the data source assignment. See "Assign data sources to Label" on page 232 for details. For long texts increase the size of the label component by pulling the anchor points with your mouse.
4. Click the **Configure filter** icon to specify filter conditions. See "Define filters for dashboard components " on page 276 for details.
5. Enter an optional component **Name** on the **Config** tab.
6. Set the **Container** properties. See "Hide or display component header and border" on page 275 for details.
7. Activate the **Auto Refresh** option to set the automatic data retrieval for the component.
8. Select a **Style** for your displayed text in the drop-down menu.
9. Select the **Alignment** of the inserted text. Default is left aligned.
10. Specify **Actions** for the component, if required. See "Specify actions for dashboard components" on page 277 for details.

A Label will be displayed with a fixed or a dynamic text.

Assign data sources to dashboard components

To display content (e.g. charts, apps or tables) in a dashboard component you have to assign a data source to the component at first.

Most dashboard components support on server side provided data sources, such as Event, BigMemory and XML data sources. The **MashZone NextGen App** component requires a MashZone NextGen app and **MashZone NextGen Views** takes only MashZone NextGen Mashups as data sources. For the **Inputfield** and **Image** components data sources are not required.

For XML, JDBC data sources and RAQL queries you can define dynamical and reusable input parameters, see "Create Input Parameters" on page 273.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.

2. Click the inserted component. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment. The **Edit data souce** dialog will be displayed.
4. Select and configure the relevant data source depending on the selected component.

The content of the selected component will be displayed with real data of the assigned data source.

Assign data sources to Line chart

To display a Line chart you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.
4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.
5. Select a data source operator in the **Add a data source** area and specify your settings.
   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.
   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.
   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.
   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.
   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.
   f. Assign **JDBC** data source.

      See "Assign JDBC data source" on page 241 for details.

6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.

   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

   b. Specify your settings.

   c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign Data columns to chart axis.

   You can assign a separate column to a chart axis as dimension (X-Axis) or KPI (Y-Axis). You can display one dimension and several KPIs or two dimensions and one KPI in the chart.

   a. Assign a **Data column** to the **X-Axis** as dimension via drag and drop.

   b. Enter a **New Column Name** and **Axis Name** if necessary.

   c. Optionally, assign a **Data column** to the **Partition** as second dimension via drag and drop. Enter a **New Column Name** and **Axis Name** if necessary.

   d. Assign one or more **Data column** to the Y-Axis as KPI via drag and drop. Enter a **New Column Name**, select a **Format** in the dop down menu and change the **Aggregation** type if necessary.

9. Edit the **Y-Axis** properties.

   a. Click on the **Y-Axis** box.

   b. Assign a axis title to the Y-Axis manually. By default, the axis title consists of the concatenated names of the assigned columns. Instead you can assign an individual name to the Y-Axis. Activate the **Manual Axis Title** option and enter a title in the **Axis Title** input field.

   c. Select an output format for the Y-Axis in the **Axis Format** selection box. The selected format overwrites the formats of the individual assigned columns. By default, the individual formats are used for tool tip display. If desired, you can add a prefix or postfix to the format, e. g. $ 1,234 or 1,234 mm.

   d. Assign a **Data Range** to the Y-Axis. By default, the complete data range of the data source is used. You can restrict the data range by setting a lower (**From**) and upper (**To**) limit. To adapt automatically the Data Range to the incoming data you can activate the **Dynamically adapt data range** option. The option adapt the limits in the case the incoming data are beyond the set limits.

10. Click **Edit Data Source** to change the source settings.

11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to Column chart

To display a Column chart you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.
4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.
5. Select a data source operator in the **Add a data source** area and specify your settings.
   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.
   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.
   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.
   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.
   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.
   f. Assign **JDBC** data source.

      See "Assign JDBC data source" on page 241 for details.
6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.
   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

b. Specify your settings.

c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign Data columns to chart axis.

   You can assign a separate column to a chart axis as dimension (X-Axis) or KPI (Y-Axis). You can display one dimension and several KPIs or two dimensions and one KPI in the chart.

   a. Assign a **Data column** to the **X-Axis** as dimension via drag and drop. Enter a **New Column Name** and **Axis Name** if necessary.

   b. Optionally, assign a **Data column** to the **Partition** as second dimension via drag and drop. Enter a **New Column Name** and **Axis Name** if necessary.

   c. Assign one or more **Data column** to the **Y-Axis** as KPI via drag and drop. Enter a **New Column Name**, select a **Format** in the drop down menu and change the **Aggregation** type if necessary.

9. Edit the **Y-Axis** properties.

   a. Click on the **Y-Axis** box.

   b. Assign a axis title to the Y-Axis manually. By default, the axis title consists of the concatenated names of the assigned columns. Instead you can assign an individual name to the Y-Axis. Activate the **Manual Axis Title** option and enter a title in the **Axis Title** input field.

   c. Select an output format for the Y-Axis in the **Axis Format** selection box. The selected format overwrites the formats of the individual assigned columns. By default, the individual formats are used for tool tip display. If desired, you can add a prefix or postfix to the format, e. g. $ 1,234 or 1,234 mm.

   d. Assign a **Data Range** to the Y-Axis. By default, the complete data range of the data source is used. You can restrict the data range by setting a lower (**From**) and upper (**To**) limit. To adapt automatically the Data Range to the incoming data you can activate the **Dynamically adapt data range** option. The option adapt the limits in the case the incoming data are beyond the set limits.

10. Click **Edit Data Source** to change the source settings.

11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to Bar chart

To display a Bar chart you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component. The relevant properies dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.
4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.
5. Select a data source operator in the **Add a data source** area and specify your settings.
   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.
   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.
   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.
   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.
   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.
   f. Assign **JDBC** data source.

      See "Assign JDBC data source" on page 241 for details.
6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.
   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.
   b. Specify your settings.
   c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.
7. Click **Next**.

8. Assign Data columns to chart axis.

   You can assign a separate column to a chart axis as dimension (X-Axis) or KPI (Y-Axis). You can display one dimension and several KPIs or two dimensions and one KPI in the chart.

   a. Assign a **Data column** to the **X-Axis** as dimension via drag and drop. Enter a **New Column Name** and **Axis Name** if necessary.

   b. Optionally, assign a **Data column** to the **Partition** as second dimension via drag and drop. Enter a **New Column Name** and **Axis Name** if necessary.

   c. Assign one or more **Data column** to the **Y-Axis** as KPI via drag and drop. Enter a **New Column Name**, select a **Format** in the dop down menu and change the **Aggregation** type if necessary.

9. Edit the **Y-Axis** properties.

   a. Click on the **Y-Axis** box.

   b. Assign a axis title to the Y-Axis manually. By default, the axis title consists of the concatenated names of the assigned columns. Instead you can assign an individual name to the Y-Axis. Activate the **Manual Axis Title** option and enter a title in the **Axis Title** input field.

   c. Select an output format for the Y-Axis in the **Axis Format** selection box. The selected format overwrites the formats of the individual assigned columns. By default, the individual formats are used for tool tip display. If desired, you can add a prefix or postfix to the format, e. g. $ 1,234 or 1,234 mm.

   d. Assign a **Data Range** to the Y-Axis. By default, the complete data range of the data source is used. You can restrict the data range by setting a lower (**From**) and upper (**To**) limit. To adapt automatically the Data Range to the incoming data you can activate the **Dynamically adapt data range** option. The option adapt the limits in the case the incoming data are beyond the set limits.

10. Click **Edit Data Source** to change the source settings.

11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to Pie chart

To display a Pie chart you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.

2. Click an inserted chart component. The relevant properies dialog is displayed.

3. Click the **Assign Data** icon to edit the data source assignment.

The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.

4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.

5. Select a data source operator in the **Add a data source** area and specify your settings.

   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.

   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.

   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.

   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.

   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.

   f. Assign **JDBC** data source.

      See "Assign JDBC data source" on page 241 for details.

6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.

   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

   b. Specify your settings.

   c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign a numeric data column to the **KPI** field.

   a. Drag and drop a numeric data column in the **KPI** field.

   b. Enter a new **Display Name** of the KPI if necessary.

   c. Change the **Aggregation** type if necessary. The aggregation type specifies how the KPI value is calculated.

d. Select a **Format** for the KPI in the drop-down menu.

e. Activate the **Round numerically** option to display rounded KPI values in the chart.

9. Assign a data column of data typ text or date to the **Partition** field.

   The assigned data column will be used as dimension for iteration of the KPI.

   a. Enter a new **Display Name** of the **Partition** if necessary.

   b. Select a **Format** in the drop-down menu. This option is only available for columns of typ date.

10. Click **Edit Data Source** to change the source settings.

11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to Grid

To display a table in a **Grid** component you have to assign a relevant data source to the component.

The **Grid** component requires an **Event data source** or **BigMemory data source**.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted **Grid** component. The relevant properies dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.

4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.

5. Select a data source operator in the **Add a data source** area and specify your settings.

   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.

   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.

   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.

   d. Assign **XML** data source.

See "Assign XML data source" on page 238 for details.

e. Assign **ARIS Table** data source.

See "Assign ARIS Table data source" on page 240 for details.

f. Assign **JDBC** data source.

See "Assign JDBC data source" on page 241 for details.

6. In addition to the data source operator you can add further operators to transform the source data.

See "Data transformation operators" on page 243 for details.

a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

b. Specify your settings.

c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign data source columns to grid columns.

You can assign one or more columns of your data source to the Grid component.

a. Insert a **Data column** in the **Grid Colums** box via drag and drop.

b. Click a grid column and enter a **New Column Name** if necessary.

c. Change the **Aggregation** type if necessary. The aggregation type specifies how the KPI value is calculated. This option is only available for columns of typ numeric.

d. Select a **Format** in the drop-down menu. This option is only available for columns of typ date and numeric.

e. Activate the **Round numerically** option to display rounded KPI values in the chart.

f. Activate the **Show negative values in red** option to display negative KPI values red highlighted.

g. Activate the **Show cell value** option to display the values of the column.

h. Activate the **Make clickable (for actions)** option to make the column clickable for triggering actions. See "Specify actions for dashboard components" on page 277 for details.

9. Click **Edit Data Source** to change the source settings.

10. Click **OK**.

The table will be displayed with real data of the assigned data source.

Configure thresholds of the Grid component

"You have already assigned data source columns to the Grid columns." on page 221.

You can define thresholds for numeric and text grid columns to display the status of a KPI. A coloured marker shows the threshold range in which a KPI value is located.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted Grid component. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon.

   The **Assign Data** dialog will be displayed if data source columns have already been assigned. Otherwise the **Edit Data Source** dialog will be displayed.
4. In the **Assign Data** dialog click on an assigned numeric or text column in the **Grid Columns** box.

   The **Text** and the **Thresholds** tabs are displayed.
5. Click on the **Thresholds** tab.
6. Click on **Background** to set KPI values and the grid background colors.

   Here you can define a KPI value range for the thresholds and the corresponding colors of the grid cells background.

   a. Click on a **Color** box to select a color for each threshold.

   b. Select an operator for each threshold to define the KPI value range, e. g. < (less than).

      The available operators are different for numeric (e.g., **< (less than)** or **= (equals to)**) and text (e.g., **starts with** or **is equal to**).

   c. Enter a value for each threshold.

      You can either enter fix threshold values or you can choose columns from your data source representing dynamic threshold values. For dynamic threshold values only text data columns are allowed. Assign the required text data column from the data source to a threshold field via drag and drop.

   d. Click on **Plus** button to add a threshold or click on **Minus** button to remove a threshold.

   e. Activate the **Colorize row** option to use the background coloring for all others cells in the row.
7. Click on **Foreground** to set KPI values and the font colors.

   Here you can define a KPI value range for the thresholds and the corresponding font colors.

a. Click on a **Color** box to select a font color for each threshold.

b. Select an operator for each threshold to define the KPI value range, e. g. < (less than).

The available operators are different for numeric (e.g., **< (less than)** or **= (equals to)**) and text (e.g., **starts with** or **is equal to**).

c. Enter a value for each threshold.

You can either enter fix threshold values or you can choose columns from your data source representing dynamic threshold values. For dynamic threshold values only text data columns are allowed. Assign the required text data column from the data source to a threshold field via drag and drop.

d. Click on ⊕ **Plus** button to add a threshold or click on ⊖ **Minus** button to remove a threshold.

e. Activate the **Colorize row** option to use the font coloring for all others cells in the row.

8. Click on **Traffic Lights** to set KPI values and to define traffic lights.

Here you can define a KPI value range for the thresholds and corresponding traffic lights. The traffic lights will be displayed in each cell of the rows. By default there are three traffic light phases plus the inactive phase. In addition of the color you can also select a different shape for each traffic light phase.

a. Click on a **Color** box to select a color and a shape for each traffic light phase.

You can use custom images instead of the colored shapes for each traffic light phase. For this activate the **Image based Traffic Light** option, click on the drop-down menu and enter an **Image URL**.

b. Activate the **Multi State** option to show multiple states of traffic lights instead of a single traffic light.

c. Select an operator for each threshold to define the KPI value range, e. g. is equal to.

d. Enter a value for each threshold.

You can either enter fix threshold values or you can choose numeric columns from your data source representing the threshold values. Assign a data column from the data source to a threshold field via drag and drop.

e. Click on ⊕ **Plus** button to add a traffic light phase or click on ⊖ **Minus** button to remove a phase.

f. Activate the **Switch position** option to place the traffic light symbol right hand of the values.

g. Activate the **Make clickable (for actions)** option to make the traffic lights clickable for triggering actions. See "Specify actions for dashboard components" on page 277 for details.

h. Activate the **Show tooltip** option to show tool tips for each traffic light symbol.

i. By default the traffic light phase is also visible if it is inactive (A KPI value is not available.). Deactivate the **Inactive state** option if the traffic light phase should not be visible. You can also change the color of the inactive traffic light phase. In case of a selected image you can only deactivate the **Show inactive state** option.

9. Click **OK**.

The defined thresholds will be displayed in the grid component.

Assign data sources to Speedometer chart

To display a Speedometer you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

The red pointer in the chart indicates the aggregated actual value of the KPI. For the displayed KPI value range you can define a minimum and a maximum value. Additionally you can compare the actual KPI value against several threshold values. The value ranges in the speedometer are indicated by different colors.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component. The relevant properies dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.

4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.

5. Select a data source operator in the **Add a data source** area and specify your settings.

   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.

   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.

   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.

   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.

e. Assign **ARIS Table** data source.

See "Assign ARIS Table data source" on page 240 for details.

f. Assign **JDBC** data source.

See "Assign JDBC data source" on page 241 for details.

6. In addition to the data source operator you can add further operators to transform the source data.

See "Data transformation operators" on page 243 for details.

a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

b. Specify your settings.

c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign a numeric data column to the speedometer pointer as aggregated KPI.

a. Drag and drop a numeric data column in the **Data column** field.

b. Enter a new **Display Name** of the KPI if necessary.

c. Change the **Aggregation** type if necessary. The aggregation type specifies how the KPI value is calculated.

d. Select a **Format** for the KPI and threshold values in the drop down menu.

e. Activate the **Round numerically** option to display rounded KPI values in the chart.

f. Enter the **Min Value** (minimum KPI value) and **Max Value** (maximum KPI value) to define the KPI value range. If the values are not set the values are calculated automatically.

9. Configure the KPI thresholds.

a. Click **Thresholds** to configure the threshold value ranges.

b. Enter a constant number in the input field or drag a numeric data column to the input field for each threshold value.

c. Select an operator (e. g., less than **<** or greater than **>**) in the drop down menu for each threshold value.

d. Click a color field and select a color for each threshold value range.

e. Click the ⊕ icon to add a threshold.

f. Click the ⊖ icon to delete a threshold.

10. Click **Edit Data Source** to change the source settings.

11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to Horizontal and Vertical Gauge Chart

To display a Horizontal or Vertical Gauge Chart you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

The pointer in the chart indicates the aggregated actual value of the KPI. For the displayed KPI value range you can define a minimum and a maximum value. Additionally you can compare the actual KPI value against several threshold values. The value ranges in the Gauge Charts are indicated by different colors.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component. The relevant properies dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.
4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.
5. Select a data source operator in the **Add a data source** area and specify your settings.
   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.
   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.
   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.
   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.
   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.
   f. Assign **JDBC** data source.

See "Assign JDBC data source" on page 241 for details.

6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.

   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

   b. Specify your settings.

   c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign a numeric data column to the speedometer pointer as aggregated KPI.

   a. Drag and drop a numeric data column in the **Data column** field.

   b. Enter a new **Display Name** of the KPI if necessary.

   c. Change the **Aggregation** type if necessary. The aggregation type specifies how the KPI value is calculated.

   d. Select a output **Format** for the KPI and threshold values in the drop down menu.

   e. Activate the **Round numerically** option to display rounded KPI values in the chart.

   f. Enter the **Min Value** (minimum KPI value) and **Max Value** (maximum KPI value) to define the KPI value range. If the values are not set the values are calculated automatically.

9. Configure the KPI thresholds.

   a. Click **Thresholds** to configure the threshold value ranges.

   b. Enter a constant number in the input field or drag a numeric data column to the input field for each threshold value.

   c. Select an operator (e. g., less than **<** or greater than **>**) in the drop down menu for each threshold value.

   d. Click a color field and select a color for each threshold value range.

   e. Click the ⊕ icon to add a threshold.

   f. Click the ⊖ icon to delete a threshold.

10. Click **Edit Data Source** to change the source settings.

11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to Traffic Light

You can use a Traffic Light to display the status of a KPI. A coloured marker shows the threshold range in which a KPI value is located.

To use a Traffic Light you have to assign a data source to the component at first. Then you select a numeric column of your data source representing the KPI and you set thresholds to define the numeric ranges of your traffic light.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted Traffic Light. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.
4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.
5. Select a data source operator in the **Add a data source** area and specify your settings.
   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.
   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.
   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.
   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.
   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.
   f. Assign **JDBC** data source.

      See "Assign JDBC data source" on page 241 for details.
6. In addition to the data source operator you can add further operators to transform the source data.

See "Data transformation operators" on page 243 for details.

a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

b. Specify your settings.

c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign Data columns to Traffic Light.

a. Assign a data column from the data source to the **Data Column** field via drag and drop.

b. Enter a **Display Name**.

c. For numeric values you can change the **Aggregation** type if necessary. Default is **Avg** (Average value).

d. Select an output **Format** in the drop-down menu or enter a format manually. The **Format** option is only available for columns of type number.

e. Activate the **Round numerically** option to round numeric values.

9. Set thresholds to define traffic light phase for KPI values.

a. Click on **Thresholds** bar.

By default there are three traffic light phases plus the inactive phase.

b. Click on **Plus** button to add a traffic light phase or click on **Minus** button to remove a phase.

c. Enter a threshold value for each traffic light phase.

You can either enter fix threshold values or you can choose numeric columns from your data source representing the threshold values. Assign a data column from the data source to a traffic light phase field via drag and drop.

d. Click on a **Shape** box to select a **Color** and a **Shape** for each traffic light phase.

e. Select an operator for each traffic light phase to define the KPI value range, e. g. < (less than).

f. You can use custom images instead of the colored shapes. Click on the drop-down menu and select **Image**. Then click on a **Shape** box and enter an URL to the relevant image.

g. By default the traffic light phase is also visible if it is inactive (A KPI value is not available.). Deactivate the **Inactive state** option if the traffic light phase should not be visible. You can also change the color of the inactive traffic light phase. In case of a selected image you can only deactivate the **Show inactive state** option.

10. Click **Edit Data Source** to change the source settings.
11. Click **OK**.

The chart will be displayed with real data of the assigned data source.

Assign data sources to MashZone NextGen App

To display a MashZone NextGen App in a **MashZone NextGen App** component you have to assign a relevant data source to the component.

The **MashZone NextGen App** component requires a MashZone NextGen App as data sources.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the inserted **MashZone NextGen App** component. The relevant properties dialog is displayed.
3. Click the **Configure** icon to edit the data source assignment.
4. Enter the relevant App ID in the **MashZone NextGen App ID** input field.
5. Click **OK**.

The MashZone NextGen App will be displayed with real data of the assigned data source.

Assign data sources to MashZone NextGen Views

To display a MashZone NextGen Mashup in a **MashZone NextGen Views** component you have to assign a relevant data source to the component.

The **MashZone NextGen Views** component requires a MashZone NextGen Mashup as data sources.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click the inserted **MashZone NextGen Views** component. The relevant properties dialog is displayed.
3. Click the **Configure** icon to edit the data source assignment.
4. Enter the relevant Mashup ID in the **MashZone NextGen View** input field.
5. Click **OK**.

The MashZone NextGen Mashup will be displayed with real data of the assigned data source.

Assign data sources to Drop-down box

To use the Drop-down box you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted Drop-down box component. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.
4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.
5. Select a data source operator in the **Add a data source** area and specify your settings.
   a. Assign **Event** data source.

      See"Assign Event data source" on page 234 for details.
   b. Assign **BigMemory** data source.

      See "Assign BigMemory data source" on page 235 for details.
   c. Assign **CSV** data source.

      See "Assign CSV data source" on page 236 for details.
   d. Assign **XML** data source.

      See "Assign XML data source" on page 238 for details.
   e. Assign **ARIS Table** data source.

      See "Assign ARIS Table data source" on page 240 for details.
   f. Assign **JDBC** data source.

      See "Assign JDBC data source" on page 241 for details.
6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.
   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

b. Specify your settings.

c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.
8. Assign a data column to the combobox.

   You can assign a separate column of the data source to the combobox. The data will be displayed in the drop down menu of the combobox.

   a. Assign a **Data column** to the **Visible Column** field via drag and drop. Enter a **New Column Name** if necessary.

   b. Select a **Format** in the drop-down menu and enter an **Initial selected value** if necessary. The **Format** option is only available for columns of type text and date.

   c. Activate the **Sort descending** option to display the values in a descending order in the drop-down menu.

9. Click **Edit Data Source** if you want to change the source settings.
10. Click **OK**.

The Drop-down box will be displayed with real data of the assigned data source.

Assign data sources to Label

To use the Label you have to assign a relevant data source to the component.

The component requires an **Event**, **BigMemory**, **CSV**, **XML**, **ARIS Table** or **JDBC** data source.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted Label component. The relevant properties dialog is displayed.
3. Click the **Assign Data** icon to edit the data source assignment.

   The **Edit Data Souce** dialog will be displayed. Here you can select a data source and create dynamic **Input Parameters**. The **Data Flow** box shows an overview of the data source configuration.

4. Optionally you can create input parameters.

   You can define dynamical and reusable input parameters for XML data sources, RAQL queries and ARIS table, see "Create Input Parameters" on page 273.

5. Select a data source operator in the **Add a data source** area and specify your settings.

   a. Assign **Event** data source.

   See"Assign Event data source" on page 234 for details.

b. Assign **BigMemory** data source.

See "Assign BigMemory data source" on page 235 for details.

c. Assign **CSV** data source.

See "Assign CSV data source" on page 236 for details.

d. Assign **XML** data source.

See "Assign XML data source" on page 238 for details.

e. Assign **ARIS Table** data source.

See "Assign ARIS Table data source" on page 240 for details.

f. Assign **JDBC** data source.

See "Assign JDBC data source" on page 241 for details.

6. In addition to the data source operator you can add further operators to transform the source data.

   See "Data transformation operators" on page 243 for details.

   a. Select an operator, e. g. **Change data type**, in the **Add data operations** area. The relevant operator will be displayed.

   b. Specify your settings.

   c. You can display a preview of the transformed source data of each operator. This enables you to track all data changes step by step. Click the **Calculate peview** icon of an operator.

7. Click **Next**.

8. Assign a data column to the label component.

   You can assign a separate column of the data source to the Label component.

   a. Assign a data column from the data source to the **Data Column** field via drag and drop.

   b. Enter a **New Column Name**.

   c. Select an output **Format** in the drop-down menu or enter a format manually. The **Format** option is only available for columns of type number and date.

   d. For numeric values you can change the **Aggregation** type if necessary. Default is **Avg** (Average value).

   e. Activate the **Round numerically** option to round numeric values.

9. Click **Edit Data Source** if you want to change the source settings.

10. Click **OK**.

The label component will be displayed with real data of the assigned data source.

Note: If you've assigned a text or date column to the component the values are taken from the first data row. Therefore in most cases the label needs to be filtered by other components.

Data source operators

To display content (e.g. charts, apps or tables) in a dashboard component you have to assign a data source to the component at first. The following data source operators are available in MashZone NextGen.

- **Event**, see "Assign Event data source" on page 234
- **BigMemory**, see "Assign BigMemory data source" on page 235
- **CSV**, see "Assign CSV data source" on page 236
- **XML**, see "Assign XML data source" on page 238
- **ARIS Table**, see "Assign ARIS Table data source" on page 240
- **JDBC**, see "Assign JDBC data source" on page 241

Furthermore, MashZone NextGen Apps and MashZone NextGen Views can be assigned as data sources to the corresponding dashboard components.

- "Assign data sources to MashZone NextGen App" on page 230
- "Assign data sources to MashZone NextGen Views" on page 230

Assign data sources to dashboard components

Assign Event data source

You can assign an **Event** data source to a dashboard component.

Procedure

1. Open the **Assign Data (1/2)** dialog of a dashboard component. See "Assign data sources to dashboard components" on page 212 for details.
2. Click the **Apama / EDA Event** button in the **Add a data source** area.
3. Select an **Event Source** in the drop-down menu.
4. Click **Next**.

The data source is assigned to the selected dashboard component.

You can display a preview of the source data. Click the **Calculate peview** icon of the operator.

In addition to the data source operator you can add further operators to transform the source data. See"Data transformation operators" on page 243 for details.

Assign BigMemory data source

You can assign a **BigMemory** data source to a dashboard component.

If you have defined one or more input parameters for the selected component the **Insert input parameter** button is available for the BigMemory operator. By clicking the button you can insert a defined input parameter at the cursor position. Click on the inserted parameter for more options. See "Create Input Parameters" on page 273.

Procedure

1. Open the **Assign Data (1/2)** dialog of a dashboard component. See "Assign data sources to dashboard components" on page 212 for details.
2. Click the **BigMemory** button in the **Add a data source** area.
3. Select a cache manager in the **Cache Alias** drop-down menu.
4. Click on the **Refresh Rate** icon and select a refresh rate in the drop-down menu. The Refresh Rate specifies the time until the source file is read in again. Default value is 30 sec.
5. Enter a **RAQL query**.

   You can enter any RAQL statement. See example **BigMemory data source** below.

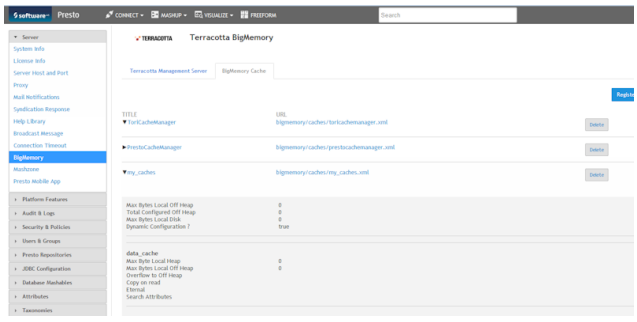
   Preset query: SELECT * FROM.
6. Click **Next**.

The data source is assigned to the selected dashboard component.

Example: **BigMemory Data source**

To use the RAQL source operator, you have to configure a BigMemory connection in MashZone NextGen in advance, see "BigMemory connection in Presto" on page 1661.

To use the RAQL BigMemory operator you have to follow these conventions. The **Cache Alias** pattern is <configuration name>.<cache name>. <cache name>. <configuration name> is the **Title** you specified as name for the configuration, <cache name> is one of the caches available in the configuration. In this example, the **Cache Alias** is "my_caches.data_cache".



In the statement field, you can enter the RAQL statement. If you refer to a cache within the configuration, you just specify the cache name (i. e. without the configuration name).

In the above example, if you wanted to address the "data_cache" cache, you would formulate your RAQL statement similar to this:

```
SELECT description FROM data_cache
```

You can display a preview of the source data. Click the **Calculate peview** icon of the operator.

In addition to the data source operator you can add further operators to transform the source data. See"Data transformation operators" on page 243 for details.

Assign CSV data source

You can assign a **CSV** data source to a dashboard component.

Procedure

1. Open the **Assign Data (1/2)** dialog of a dashboard component. See "Assign data sources to dashboard components" on page 212 for details.
2. Click the **CSV** button in the **Add a data source** area.
3. Enter the URL to the CSV file in the **Source** field.
4. If required, you can upload CSV files to the MashZone NextGen Repository. For this, click on **Upload a file** to open the **File Resources** administration area of the **Admin Console**.

   See "Add External Resources as MashZone NextGen Files" on page 1746 for details.

   To upload a file to the MashZone NextGen Repository administration privileges are required.

   If you use an absolute URL, e.g. `http://anyhost:8080/presto/files/admin_csv?1454486242000`, then an authentication has to be set. All requests to the uploaded file will be made with the specified user / password combination.

   If you us a relative URL, e.g. `/presto/files/admin_csv?1454486242000`, then an authentication is not required. All requests to the uploaded file will be made with the current logged in user.

   In some cases, it might be necessary to grant view permission for a user. This can be done with the API console, e.g.:

```
{
"version":"1.1",
"sid":"PolicyService",
"svcVersion":"0.1",
"oid":"addPermissions",
"params":[
"fileName","type.entity.file","VIEW",
[{"principalId":"userID","principalTypeId":"User"}]]
}
```

   - `fileName` is the name that was specified while uploading the file.
   - `userID` is the id of the user, who should get the permission.

5. Click the **Refresh Rate** icon and select a Refresh Rate in the drop-down menu. The Refresh Rate specifies the time until the source file is read in again. Default value is 30 sec.
6. You can optionally add HTTP headers to the URL. Click **HTTP Headers** and select a header in the drop-down menu and insert a parameter in the input field.
7. You can optionally add a HTTP basic access authentication. Click **Authentication** and enter a user name in the **User** input box. Enter the **Password** of the user and click **Save Changes**.
8. Select a **Separator** in the drop-down menu. A Separator separates the column values in the CSV file. Available are comma (,), semicolon (;), space, tab and pipe ((l)), default is comma.
9. Activate the **Get column names from source** option and enter the number of the relevant row. The number specifies a particular row that contains the column names.
10. Enter a number in the **Data from row** input field. The number specifies a particular row from which the data source values extraction starts.
11. Set the **Advanced parsing options**.
    - **Charset**: Character set in which the source file is coded. The default value can be changed manually if the extracted data refers to a different coding type. Default: windows-1252.
    - **Masking**: Protects the enclosed characters against being split at the separator. If column values contain the specified separator, they can be enclosed in a pair of masking characters, e. g., "1,23". Masking characters can be set as required (available masking characters: single-quote ('), double-quote ("), none).
    - **Remove quote characters in column value**: Removes the characters used for masking from the result data. If this option is deactivated, then the masking characters will remain as part of the result data.
    - **Sanitize Names**: Transforms column names such that they can be used as XML names, in the same way as EMML treats them. This affects names containing spaces or other special characters. If this option is deactivated, then the names are kept as is.
    - **Trim Whitespace**: Removes all leading and trailing whitespace from column values. If this option is deactivated, then any whitespace will be part of the result data.
    - **EMML Parsing**: Parses values in the same way as they are parsed in EMML. This affects parsing of numeric and date values. It affects whether a particular value is understood as a date or numeric value, because different sets of date patterns and locales are used. If this option is deactivated, then dates without an explicit time zone are assigned to the server's default time zone, while when this option is deactivated, they are assigned GMT.
12. You can edit the list of requested columns of the data source. Click **Columns: Detect** to read out the columns of the data source automatically. The requested columns list will be displayed. Click **Edit** to configure the columns list. Unselect a column for

excluding it from the result data. Entering a new name for a column will cause that to be used instead of the original column name in the result. Activating the **Keep as Text** option will force the column's values to be parsed as text rather than numeric or date. Click **OK** to save your changes.

13. Click **Next**.

The data source is assigned to the selected dashboard component.

You can display a preview of the source data. Click the **Calculate peview** icon of the operator.

In addition to the data source operator you can add further operators to transform the source data. See"Data transformation operators" on page 243 for details.

Assign XML data source

You can assign a **XML** data source to a dashboard component.

If you have defined one or more input parameters for the selected component the **Insert input parameter** button is available for the **XML** operator. By clicking the button you can insert a defined input parameter at the cursor position. Click on the inserted parameter for more options. See "Create Input Parameters" on page 273.

Procedure

1. Open the **Assign Data (1/2)** dialog of a dashboard component. See "Assign data sources to dashboard components" on page 212 for details.
2. Click the **XML** button in the **Add a data source** area.
3. Enter the URL to the XML file in the **Source** field.

   You can also enter MashZone NextGen Mashups and Mashables in the field. As soon as you start typing, all supported MashZone NextGen Mashups and Mashables will be auto suggested. By selecting one of them, the URL of that artifact will be set.
4. If required, you can upload XML files to the MashZone NextGen Repository. For this, click on **Upload a file** to open the **File Resources** administration area of the **Admin Console**.

   See "Add External Resources as MashZone NextGen Files" on page 1746 for details.

   To upload a file to the MashZone NextGen Repository administration privileges are required.

   If you use an absolute URL, e.g. `http://anyhost:8080/presto/files/admin_xml?1454486242000`, then an authentication has to be set. All requests to the uploaded file will be made with the specified user / password combination.

   If you us a relative URL, e.g. `/presto/files/admin_xml?1454486242000`, then an authentication is not required. All requests to the uploaded file will be made with the current logged in user.

   In some cases, it might be necessary to grant view permission for a user. This can be done with the API console, e.g.:

```
{
"version":"1.1",
"sid":"PolicyService",
"svcVersion":"0.1",
"oid":"addPermissions",
"params":[
"fileName","type.entity.file","VIEW",
[{"principalId":"userID","principalTypeId":"User"}]]
}
```

- `fileName` is the name that was specified while uploading the file.
- `userID` is the id of the user, who should get the permission.

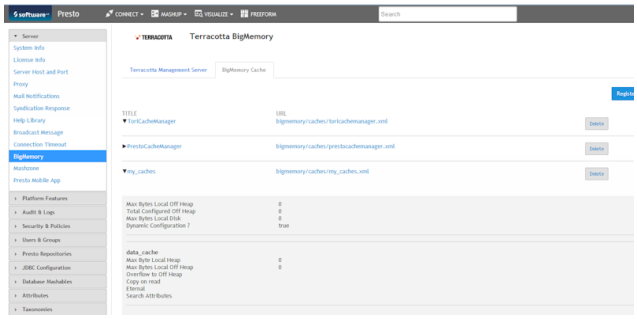
5. Click on the **Refresh Rate** icon and select a refresh rate in the drop-down menu. The Refresh Rate specifies the time until the source file is read in again. Default value is 30 sec.
6. You can optionally add HTTP headers to the URL. Click **HTTP Headers** and select a header in the drop-down menu and insert a parameter in the input field.
7. You can optionally add a HTTP basic access authentication. Click **Authentication** and enter a user name in the **User** input box. Enter the **Password** of the user and click **Save Changes**.
8. Specify the **Repeating element** of your data source in the corresponding input box. You can click **Repeating element: Detect** to read out the repeating element automatically and edit it in the input box. A repeating element already set by a user will not be considered. The repeating element will be displayed corresponding to the hierarchy of the xml elements, e. g. *<element>/<repeating element>* . Depending on the repeating element, the requested columns will be displayed in the **Columns** box.
9. You can edit the list of requested columns of the data source.
   a. Click **Columns: Detect** to read out the columns of the data source automatically. The requested columns list will be displayed, based on the specified repeating element.
   b. Click **Edit** to configure the columns list.
   c. You can edit a column name in the **Name** input field.
   d. You can assign a new **Data type** to a column.
   e. You can edit a column path in the **Path** input field.
   f. Click the **Delete** icon to remove a column from the list.
   g. Click **Add new colum** to add a new data source column.
   h. Click **Copy to clipboard** to copy the column configuration in text form to clipboard.
   i. Click **Insert new colum config** to enter a column configuration in text form. If you click **OK** then all column configurations will be overwritten.
   j. Click **Ok** to save your changes.
10. Click **Next**.

The data source is assigned to the selected dashboard component.

Example: **BigMemory Data source**

To use the RAQL source operator, you have to configure a BigMemory connection in MashZone NextGen in advance, see "BigMemory connection in Presto" on page 1661.

To use the RAQL BigMemory operator you have to follow these conventions. The **Cache Alias** pattern is <configuration name>.<cache name>. <cache name>. <configuration name> is the **Title** you specified as name for the configuration, <cache name> is one of the caches available in the configuration. In this example, the **Cache Alias** is "my_caches.data_cache".



In the statement field, you can enter the RAQL statement. If you refer to a cache within the configuration, you just specify the cache name (i. e. without the configuration name). In the above example, if you wanted to address the "data_cache" cache, you would formulate your RAQL statement similar to this:

```
SELECT description FROM data_cache
```

You can display a preview of the source data. Click the **Calculate peview** icon of the operator.

In addition to the data source operator you can add further operators to transform the source data. See"Data transformation operators" on page 243 for details.

Assign ARIS Table data source

You can assign an **ARIS Table** data source to a dashboard component.

This data source operator extracts data from an ARIS table. In ARIS, you can export a model of the Table type and generate a link to the exported file in the form of an URL. Detailed information on how to create an ARIS table is available in the ARIS online help.

If you have defined one or more input parameters for the selected component the **Insert input parameter** button is available for the **ARIS Table** operator. By clicking the button you can insert a defined input parameter at the cursor position. Click on the inserted parameter for more options. See "Create Input Parameters" on page 273.

Procedure

1. Open the **Assign Data (1/2)** dialog of a dashboard component. See "Assign data sources to dashboard components" on page 212 for details.

2. Click the **ARIS Table** button in the **Add a data source** area.
3. Enter the URL to the ARIS Table file in the **Source** field.
4. Click on the **Refresh Rate** icon and select a refresh rate in the drop-down menu. The Refresh Rate specifies the time until the source file is read in again. Default value is 30 sec.
5. You can optionally add HTTP headers to the URL. Click **HTTP Headers** and select a header in the drop-down menu and insert a parameter in the input field.
6. You can optionally add a HTTP basic access authentication or you can use an existing authentication defined in the MashZone NextGen administration. Click **Authentication** and enter a user name in the **User** input box. Enter the **Password** of the user and click **Save Changes**.
7. Click **Parameters: Detect** to read out potential parameters of the data source automatically. A requested parameters list will be displayed, based on the specified ARIS Table source. You can enter the required parameter values to read in the data source.
8. Click **Next**.

The data source is assigned to the selected dashboard component.

You can display a preview of the source data. Click the **Calculate peview** icon of the operator.

In addition to the data source operator you can add further operators to transform the source data. See "Data transformation operators" on page 243 for details.

Assign JDBC data source

You can assign a **JDBC** data source to a dashboard component.

If you have defined one or more input parameters for the selected component the **Insert input parameter** button is available for the **JDBC** operator. By clicking the button you can insert a defined input parameter at the cursor position. Click on the inserted parameter for more options. See "Create Input Parameters" on page 273.

Procedure

1. Open the **Assign Data (1/2)** dialog of a dashboard component. See "Assign data sources to dashboard components" on page 212 for details.
2. Click the **JDBC** button in the **Add a data source** area.
3. Select a JDBC database connection in the **Data Source** drop-down menu.

   JDBC database connections can be defined in the MashZone NextGen**Admin Console**. See "Manage Data Sources and Drivers" on page 1679.
4. If required, you can upload XML files to the MashZone NextGen Repository. For this, click on **Upload a file** to open the **File Resources** administration area of the **Admin Console**.

See "Add External Resources as MashZone NextGen Files" on page 1746 for details.

To upload a file to the MashZone NextGen Repository administration privileges are required.

If you use an absolute URL, e.g. `http://anyhost:8080/presto/files/admin_xml?1454486242000`, then an authentication has to be set. All requests to the uploaded file will be made with the specified user / password combination.

If you us a relative URL, e.g. `/presto/files/admin_xml?1454486242000`, then an authentication is not required. All requests to the uploaded file will be made with the current logged in user.

In some cases, it might be necessary to grant view permission for a user. This can be done with the API console, e.g.:

```
{
"version":"1.1",
"sid":"PolicyService",
"svcVersion":"0.1",
"oid":"addPermissions",
"params":[
"fileName","type.entity.file","VIEW",
[{"principalId":"userID","principalTypeId":"User"}]]
}
```

- `fileName` is the name that was specified while uploading the file.
- `userID` is the id of the user, who should get the permission.

5. Click on the **Refresh Rate** icon and select a refresh rate in the drop-down menu. The Refresh Rate specifies the time until the source file is read in again. Default value is 30 sec.
6. You can optionally add HTTP headers to the URL. Click **HTTP Headers** and select a header in the drop-down menu and insert a parameter in the input field.
7. You can optionally add a HTTP basic access authentication or you can use an existing authentication defined in the MashZone NextGen administration. Click **Authentication** and enter a user name in the **User** input box. Enter the **Password** of the user and click **Save Changes**.
8. Enter a **SQL query**.

   To have access to columns of the table it is necessary to add a SQL query command in the **SQL Query** input field, e. g. SELECT * FROM <table name>.
9. Click **Next**.

The data source is assigned to the selected dashboard component.

You can display a preview of the source data. Click the **Calculate peview** icon of the operator.

In addition to the data source operator you can add further operators to transform the source data. See"Data transformation operators" on page 243 for details.

Data transformation operators

In addition to the data source operators you can add further operators to transform the source data of dashboard components. The following data transformation operators are available in MashZone NextGen.

Assign data sources to dashboard components

Change data type

Behavior

Changes the data types of the specified columns to the **Number**, **Text**, or **Date** data types

Action	Result
Conversion of **Text** to **Number**	Numerical value of the text taking into account the decimal separator. If the decimal separator is set correctly, any thousands separator is detected automatically.
Conversion of **Number** to **Text**	Text representation of the number in the internal format, or based on the language and the specified format. You can also specify a valid number of leading zeros. If nothing is specified here, the results are formatted in the numerical format.
Conversion of **Text** to **Date**	Date value of the text in the internal format, based on the specified format and, where applicable, the language. The date must be in the AD era. Date values before the common era are not supported. The time format must be specified. The time format is made up of sequences of characters, which stand for date fields, e.g., year, month, day of the week, or minute, in the relevant language; separated by separators. In addition, the corresponding language must be specified. Non-editable text must be enclosed in quotation marks. When using the **Q** or **q** symbol for quarters, all other symbols except **Y** and **y** for years are ignored. Only the order of **Q/q** and **Y/y** is relevant.
Conversion of **Date** to **Text**	Text representation of the date in the internal format, or based on the language and the specified format Non-editable text must be enclosed in quotation marks.

Action	Result
	The format and language specifications are optional. If no format is specified, the data is output in the internal date format. If no language is specified, English (EN) is applied as the default language.
Conversion of **Number** to **Date**	Date value corresponding to the value of the number as milliseconds since 01/01/1970
Conversion of **Date** to **Number**	Number of milliseconds since 01/01/1970

Internal number format

If the user is logged in in English, the number format is Anglo-Saxon style with a period as the decimal separator and at least one decimal place, but without grouping characters.

Internal date format

yyyy-Q for specifying to the nearest quarter, otherwise yyyy-MM-ddThh:mm:ss. The number of digits corresponds to the accuracy of the date, and the remaining digits are omitted. This is the transfer format.

Quarterly specifications

These are indicated by a **Q** within the section of the format that is not in single quotation marks.

Prerequisites for conversion of text into quarterly date values:

1. It is expected that a source value containing a quarterly date consists of just two sequences of figures indicating the year and the quarter. Any non-numerical characters can occur before, after and between them, e.g., Quarter 04/2009.
2. The pattern uses **Y** or **y** as the symbol for the year and **Q** or **q** for the quarter, e.g., quarter Q/y or Q Y.

Procedure:

1. The (first) two sequences of digits are determined from the source value.
2. The section of the format that is not enclosed between single quotation marks is used to determine whether **q / Q** or **y / Y** appears first.
3. If q or Q appears first, the first sequence of digits is interpreted as the quarter and the second as the year, otherwise the reverse.

Parameter

The following parameters are available.

Parameter	Description
Column	Name of the column to be changed; Source: Source table Data type: Date, Number, or Text Specification: Mandatory
New type	New column data type Default value: Text Specification: Mandatory
Format (Date type)	Time format for conversion from **Date** type to **Text** type and vice versa. The following formatting symbols are available when converting date into text: ■ Year: y or Y ■ Quarter: Q ■ Month: M ■ Calendar week: w ■ Day of the week: E or e ■ Day of the month: d ■ Day of the year: D ■ Hour: H or h ■ Minute: m ■ Second: s ■ AM/PM: a ■ Time zone: z (e.g., GMT) ■ RFC time zone: Z (e.g., -0900) ■ Era: G (must always be AD) ■ Default value: MM/dd/yyyy

Parameter	Description
	The following formatting symbols are available when converting text to date: ■ Year: y ■ Quarter: Q ■ Calendar week: w ■ Week of the month: W ■ Day of the week: E ■ Day of the month: d ■ Day of the year: D ■ Hour (0-23): H ■ Hour (1-24): k ■ Hour AM/PM (0-11): K ■ Hour AM/PM (1-12): h ■ Minute: m ■ Second: s ■ AM/PM: a ■ Time zone: z ■ RFC time zone: Z ■ Era: G Default value: yyyy-MM-dd'T'HH:mm:ss; Permitted separators in both cases: Dash/minus (-), underscore (_), slash (/), period (.), colon (:), comma (,), tab character, and space. Specification: Mandatory
Language	Language if the target format is of the **Date** type. Available languages: de and en. Specification: Mandatory when using names of months and names of days of the week
Decimal separator	Separator for the decimal places, if the target format is of the **Number** type.

Parameter	Description
	Default value: Comma (,) Specification: Mandatory
Format (**Number** source format)	Number format for the conversion of the Number type to the Text type. You can select predefined formats or set your own format manually. With manual entry, the numbers before the decimal separator must have four digits ascending and then descending, e.g., 1,234.321. After this, you can add text (such as the unit "hours" or km/h). Default value: 1234 Permitted separators: ■ Thousands separator in German: period (.) ■ Thousands separator in English: comma (,) ■ Decimal separator in German: comma (,) ■ Decimal separator in English: period (.)
Leading zeros	Number of leading zeros. The maximum number of leading zeros is the number of digits before the decimal separator. Example: Format: 1,234.12 and leading zeros: 5 Number -> Text 10,245 -> 00010,25 12000,4 -> 12000,4 89,7 -> 00089,70
Specify type	Automatically specifies the data type of the source values. If the content of a column does not correspond to its assigned data type, a row is created in the operator which specifies the data type determined for this column Vice-versa, settings (lines) are removed from the operator, which would reset the data type of a column already typified correctly.

The characters in the time format can be combined in any order and repeated any number of times.

Exceptions:

- For a month, the number of characters must be >= 3 (MMM or MMMM) if the month is specified in text format (JAN, FEB, etc.) and <3 if it is specified as a figure. In this case, a language must also be specified so that the name of the month can be transformed correctly.
- For a year format such as 2009, y can be specified any number of times, i.e., yy and yyyy return 2009.
- For a year format such as 09, however, yyyy returns the year 9 and yy the year 2009.
- When formatting date values as days of the week for a date to text conversion, an e/E number < 4 returns the day abbreviations (MON, TUE, etc.), while e/E = or > 4 returns the full name of the day.
- Only the month (M), minute (m), time zone (z), RFC time zone (Z) and week of the year (w) are case-sensitive.
- When converting text to date, if the Q or q symbol is used for quarters all other symbols except Y and y are ignored. Only the order of Q/q and Y/y is decisive then.
- For the reverse conversion from date to text, the Q/q can be combined with any other symbols, but may only occur once (not QQ/yy)
- Quarter entries are currently only possible in the form YYYY-Q or YYYY-QQ. These strings may only consist of the year, separator, and quarter.
- All other strings must be enclosed in single quotation marks ('). Spaces can be inside or outside, e.g., 'On' dd.MM.yy 'at' hh:mm, or 'On 'dd.MM.yy' at 'hh:mm' '.
- The space pattern in the source and target format must match, e.g., "2 .3 .09" -> "d .M .y" but not "2. 3. 09" -> "d .M .y".

Error behavior

The following error behavior can occur.

Error situation	Behavior
Text -> date: Time format contains characters that are not indicated	The operator returns no data. Error message

Error situation	Behavior
by quotation marks and are not available as date characters in the transferred language.	
Text -> date: Column contains non-empty strings that do not match the time format.	The incorrect cell remains empty in the target column. The data type change for the column is continued for the remaining rows. Warning message.
Text -> date: Column contains empty cells	The incorrect row remains empty. The data type change for the column is continued

Error situation	Behavior
	for the remaining rows. Warning message.
Text -> number: Column contains strings that cannot be directly interpreted as numbers, e.g., because they contain units.	Tolerance: All non-numerical characters are ignored and a numerical value is created from the rest. Exception: If, e, E, +e, +E, or -E is embedded in figures, they are interpreted as part of "computerized scientific notation" of the number and are not ignored. No message is output. Error: If the string is

Error situation	Behavior
	still not successfully changed, the affected cell remains empty. Warning message
Number -> date: The source number has decimal places.	The number is rounded to a value without decimal places and the data type is changed using this value.

Examples

Source format: "22.3.2009"

Time format: "d.M.y" or "DDDD.MM.YYYY",

but not "DD.MMM.YYYY "

Source format:"03/22/09 30:24 PM"

Time format: "MM/DD/YY hh:mm a" or "M/d/y HH:mm A"

but not "M/d/y HH:MM A" or "m/d/y HH:mm A"

Source format: "Time: 2009-FEBRUARY-01T22:33:44"

Time format: "Time: 'y-MMM-d'T'h:m:s" or "'Time:' y-MMMMM-d'T'h:m:s",

but not "'Time: 'y-MM-d'T'h:m:s"

Source format: "3. quarter 2009"

Time format: "QY" or "Q'. quarter' y or "QQ/yyyy";

but not "YQ"

Filter rows

Behavior

Filters the data feed one row at a time using particular conditions.

Column values of the Number, Text, or Date type are either let passed or blocked. An appropriate filter criterion can be selected depending on the data type.

Action	Result
Action	Executed if particular conditions are met. Possible actions: ■ Let values pass (from source table) ■ Block values (from source table) if ■ all conditions are met ■ one condition is met Default value: Let values pass if all conditions are met. Specification: Mandatory
Column	Name of the column whose values are filtered. Source: Source table Data type: Number, Text, Date; Specification: Mandatory
Comparison operator	Operator that compares the values from the source column with the comparison values. Available comparison operators depend on the data type of the source column. Default value: is equal to Specification: Mandatory
Comparison values	Values that are compared with the values from the source column.

Action	Result
	Source: Source table, single-value operator, user input, or a constant Data type: Must be identical to that of the source column. Comparison value missing ■ Condition met: If a comparison value is missing, the condition is assumed to be met. ■ Condition not met: If a comparison value is missing, the condition is assumed to not be met. Specification: Mandatory

Comparison operators

The following comparison operators are available.

Parameter	Description
Figure	Is equal to Is not equal to Is less than Is less than or equal to Is greater than Is greater than or equal to Is empty Is not empty
Text	Is equal to Is not equal to Starts with Ends with Contains Does not contain Is empty Is not empty

Parameter	Description
Date	Before After In Before or on On or after Is empty Is not empty

Error behavior

The following error behavior can occur.

Error situation	Behavior
The comparison value type does not match the column type.	The operator returns no data. Error message.

Conditional replace

Behavior

Changes the value in the specified column one row at a time if certain conditions are met.

Replaces existing values in the column with new values. Replacement must be linked to a condition, i.e., you can specify whether all or at least one condition must be met. Several conditions can be specified and these are linked to each other with "AND".

Parameter

The following parameters are available.

Action	Result
Column	Name of the column whose values are replaced. Source: Source table Data type: Number, Text, Date; Specification: Mandatory
New value	Value that replaces the value in the source column.

Action	Result
	Source: Column with values, single value from a feed (single-value operator), user input, or a constant. Default value: is equal to Specification: Mandatory
Replace	Values are replaced if one or all conditions is/are met.
Source column	Name of the column whose values are compared. Source: Source table Data type: Number, Text, Date; Specification: Mandatory
Comparison operator	Operator that compares the values from the source column with the comparison values. Available comparison operators depend on the data type of the source column. Default value: is equal to Specification: Mandatory
Comparison values	Values that are compared with the values from the source column. Source: Column with values, single value from a feed (single-value operator), user input, or a constant. Data type: Must be identical to that of the source column. Comparison value missing ■ Condition met: If a comparison value is missing, the condition is assumed to be met. ■ Condition not met: If a comparison value is missing, the condition is assumed to not be met. Specification: Mandatory

Comparison operators

The following comparison operators are available.

Data type	Comparison operators
Figure	Is equal to Is not equal to Is less than Is less than or equal to Is greater than Is greater than or equal to Is empty Is not empty
Text	Is equal to Is not equal to Starts with Ends with Contains Does not contain Is empty Is not empty
Date	Before After In Before or on On or after Is empty Is not empty

Error behavior

The following error behavior can occur.

Error situation	Behavior
New value does not match the data type of the source column.	The operator returns no data. Error message.
Comparison value in condition, value does not match the data type of the condition column.	The operator returns no data. Error message.

Arithmetic

Behavior

Executes various arithmetical calculations. The operator sets any number of numerical operands against each other. The values are set against each other one row at a time according to the specified calculation type. The calculation always runs from top to bottom. In other words, two operands are always set against each other one row at a time and the result from the first two operands is then set against the third operand.

Compounding can be mapped by using a separate operator for each expression in brackets.

Parameter

The following parameters are available.

Action	Result
Operands	One numerical operand per operation for the "Square" and "Square root" calculation types, otherwise two numerical operands.

Action	Result
	Source: Source table, constants, user input or incoming values from other operators. Data type: Number Specification: Mandatory
Calculation type	
Addition (+)	Adds two columns row by row
Subtraction (-)	Subtracts 2 columns row by row
Division (/)	Divides the first column by the second column
Multiplication (*)	Multiplies two columns row by row
Percent (%)	Multiplies row by row the second column with the percent value of the first column
Square (x^2)	Calculates the square of a column
Root	Calculates the square root of a column
Sine (sin)	Calculates for a column the sine value of an angle in degrees
Cosine (cos)	Calculates for a column the cosine value of an angle in degrees
Tangent (tan)	Calculates for a column the tangent value of an angle in degrees
Arcsine (asin)	Calculates for a column the arcsine of an angle in degrees

Action	Result
Arccosine (acos)	Calculates for a column the arccosine of an angle in degrees
Arctangent (atan)	Calculates for a column the arctangent of an angle in degrees
Logarithm (lg)	Calculates for a column the common logarithm
Logarithm (ln)	Calculates for a column the natural logarithm
Power (exp)	Calculates for a column S1 the S2nd power of S1 (S1 to the power of S2)
Minimum (min)	Calculates the minimum of column 1 and column 2
Maximum (max)	Calculates the maximum of column 1 and column 2
Absolute value (abs)	Calculates for a column the absolute value Default value: Addition (+) Specification: Mandatory You need to specify the source values for the trigonometric functions sin, cos, tan, asin, acos, and atan in degrees.

Example

Example: Result = Column 1 + Column 2 -– Column 3

Column 1	Column 2	Result Column 3
1000	2000	2950
2000	3000	4000
3000	4000	5200

Move date

Behavior

Moves a date by a specified amount of time in a given direction and writes the results to a target column.

A date can only be moved by an amount of time whose unit is the same as or less accurate than the unit of the date itself. If the format of the moving period is more accurate than the format of the source date, the source date is retained. If you move a date by quarters, it is moved by three months for every quarter.

If a date accurate to the nearest day with a number of days > 28 is moved to a month that has fewer days, the result is the last day of the target month.

Example

You can move a date accurate to the nearest month by months, quarters or years, but not by days. A date accurate to the nearest year can only be moved by years, a date accurate to the smallest unit by any unit.

Parameter

The following parameters are available.

Action	Result
Source column	Name of the source column whose date values are moved. Source: Source table Data type: Date Specification: Mandatory
Direction	Direction in which the date is moved. Valid values: Forward or Back Data type: Text Default value: Forward Specification: Mandatory
Value	Value by which the date is moved by the selected unit. Source: Manual entry or single value and source table Data type: Number Default value: 1 Specification: Optional

Action	Result
Unit	Unit of time by which the date is moved. Data type: Text Default value: Second Specification: As source The information As source allows a move even if the unit of the date values is unknown at the time of creation or if it can vary.
Target column	Name of the target column to which the result is written. Data type: Date Default value: Result_1 Specification: Optional The target column can be identical to the source column. The values in the target column are overwritten.

Error behavior

The following error behavior can occur.

Error situation	Behavior
The source column is not of the Date type.	The operator returns no data. Error message.
A row in the source column does not contain a value.	The affected rows are ignored.
A date in the source	The source

Error situation	Behavior
column is less accurate than the amount to be moved.	value is retained.
A required parameter is missing.	The operator returns no data. Error message.

Example

Source value	Value	Unit	Direction	Result
2009-12-24T16:23	10	YEAR	NEXT	2019-12-24T16:23
2009-12-24	10	DAY	NEXT	2010-01-03
2009-12-30	10	MONTH	BACK	2009-02-28
2009-11	1	QUARTER	NEXT	2010-02
2009-11	1	DAY	NEXT	2009-11
2009-01-01	3	As source	NEXT	2011-01-04
2011-Q1	3	As source	NEXT	2011-Q4

Round date

Behavior

Converts date values from a date column to a rougher time unit and writes the results to a target column.

Parameter

The following parameters are available.

Action	Result
Source column	Name of the source column whose values are rounded. Source: Source table Data type: Date Specification: Mandatory
Accuracy	Accuracy of the new date format, defined by the unit: Year, Quarter, Month, Day, Hour, Minute, or Second, and Interval: Depending on the selected unit, e.g., 5 minutes or 1 year Data types: Numeric, Text Default values: 1, Minute If the accuracy of the source column is less accurate or the same as the target column format, the original value is retained. The date values are rounded according to the selected interval. Only the unit to be rounded is taken into account, e.g., when rounding to minutes, the seconds are ignored. Rounding type: Specifies how the selected time interval is to be rounded. ■ Round up for half an interval: Automatically rounds up above an interval value higher than or equal to half of the interval value ■ Round down for half an interval: Automatically rounds down below an interval value lower than or equal to half of the interval value ■ Always round up: Always rounds up, regardless of the interval value ■ Always round down: Always rounds down, regardless of the interval value
Target column	Name of the target column to which the converted date is written Data type: Date Default value: Result_1

Action	Result
	Specification: Optional The target column can be identical to the source column. The values in the target column are overwritten. If the target column is not of the Date type, it is replaced by a new date column.

Error behavior

The following error behavior can occur.

Error situation	Behavior
The source column is not of the Date type.	The operator returns no data. Error message.
A row in the source column does not contain a value.	The affected rows are ignored.

Examples

Source value	Accuracy	Result
2009-12-24T16:23	Day	2009-12-24
2009-12-24T16:23	Hour	2009-12-24T16
2009-12-24	Month	2009-12
2009-12-24	Quarter	2009-Q4

Source value	Accuracy	Result
2009-12-24	Year	2009
Rounding		
2010-08-06T17:15:**27**	15 seconds	2010-08-06T17:15:**30**
2010-08-06T17:**17**:00	15 minutes	2010-08-06T**17**:00
2010-08-06T17:**18**:00	15 minutes	2010-08-06T**17:15**
2010-08-06T**02:48:04**	4 hours	2010-08-06T0**4:00:00**

Round up

Source value	Accuracy	Result
2010-02-28T23:**07:00:00 AM**	15 minutes	2010-02-28T**23:00**
2010-02-28T23:**07:30**	15 minutes	2010-02-28T**11:15:00 PM**
2010-**02-28**T23:30:00	1 hour	2010-**03-01**T00

Round down

Source value	Accuracy	Result
1970:01:01T09:00:**01**	6 hours	1970:01:01T**12**:00:00
1970:01:01T09:00:**00**	6 hours	1970:01:01T**06**:00:00

Always round up

Source value	Accuracy	Result
2010-02-28T**11:15:00 PM**	5 minutes	2010-02-28T**11:15:00 PM**
2010-02-28T**11:15:01 PM**	5 minutes	2010-02-28T**11:30:00 PM**
2010-**02-28T20:00:01**	6 hours	2010-**03-01**T00

Always round down

Source value	Accuracy	Result
2010-02-28T**11:15:00 PM**	5 minutes	2010-02-28T**11:15:00 PM**
2010-02-28T**11:14:59 PM**	5 minutes	2010-02-28T**23:00**
2010-**03-01T05:59:59**	6 hours	2010-**03-01T00**

Aggregate

Behavior

Combines rows if identical values occur multiple times in specified dimension columns. The numerical values in the remaining columns are combined using Average, Sum, Minimum, Maximum, or Number.

One or more numerical columns in a table are aggregated using no, one or several dimension columns. In all rows that have the same values in all dimension columns, the values in the columns to be aggregated are combined into one row based on an aggregation rule, i.e., the result contains one row for each combination of dimension columns. This also applies if no columns are specified for aggregation. If no dimension column is specified, only a single row is created and all values in the columns to be aggregated are combined to a single value for each column. No rows are created for combinations that do not occur in the original table.

Parameter

The following parameters are available.

Action	Result
Dimension column	Name of dimension column. Source: Source table Data type: Date, Number, or Text Default value: {None} Specification: Optional
Aggregation column	Name of the column to be aggregated. Source: Aggregation column is transferred from the source table. Data type: Number Default value: {None} Specification: Optional
Aggregation type	Aggregation type for the column to be aggregated: Average value, Sum, Minimum, Maximum, Count, First row and Last row. Default value: Average value, if aggregation column selected. Specification: Mandatory, if aggregation column selected.
Weighting	If the aggregation type is Average value or Sum, a numerical column for weighting of the rows can be specified for each column to be aggregated. Specification: Optional

Aggregation type

The following aggregation operations can be applied to the columns to be aggregated.

Data type	Comparison operators
Minimum	Finds all rows that have a particular combination of values in the dimension columns and returns the lowest value that occurs in these rows in the column to be aggregated.

Data type	Comparison operators
Maximum	Finds all rows that have a particular combination of values in the dimension columns and returns the highest value that occurs in these rows in the column to be aggregated.
Average value	Finds all rows that have a particular combination of values in the dimension columns and returns the average of the values in the column to be aggregated. For weighting purposes, an additional column can be specified for each source column, containing a weighting factor for each row. The weighting information is combined as a pair with the source column.
Sum	Finds all rows that have a particular combination of values in the dimension columns and returns the sum of the values in the column to be aggregated. For weighting, an additional column can be specified for each source column, containing a weighting factor for each row. The weighting information is combined as a pair with the source column.
Count	Finds all rows that have a particular combination of values in the dimension columns and returns the count of values in the column to be aggregated.
First row	Finds all rows that have a particular combination of values in the dimension columns and returns the value of the row with the lowest row index (according to the index column).
Last row	Finds all rows that have a particular combination of values in the dimension columns and returns the value of the row with the highest row index (according to the index column).

At least one dimension or aggregation column, or both, must be set.

If no aggregation columns or dimension columns are specified, the incoming table remains unchanged.

Error behavior

The following error behavior can occur.

Error situation	Behavior
The columns to be	The operator

Error situation	Behavior
aggregated are not all numerical.	returns no data. Error message.

Examples

The following table is to be aggregated based on the **Dim 1** and **Dim 2** columns.

Dim 1	Dim 2	Values 1	Weighting (values 2)
A	X	1	3
B	Y	3	4
C	Z	5	6
A	X	7	4
B	Y	9	10
C	Z	11	12

The sum is to be calculated for the Values 1 column and the average for the Values 2 column. The Weighting (values 2) column is used for weighting the Values 2 column one row at a time.

Result

Dim 1	Dim 2	Sum (values 1)	Average (values 2)
A	X	8 (1+7)	5,43 (2*3 + 8*4)/ (3+4)
B	Y	12 (3+9)	6,57 (4*4

Dim 1	Dim 2	Sum (values 1)	Average (values 2)
			+ 10*3)/ (4+3)
C	Z	16 (5+11)	9,43 (6*3 + 12*4)/ (3+4)

Insert column

Behavior

Inserts new columns of Text, Number, or Date data type into the data feed. Each of the columns can be populated with an initial value.

Parameter

The following parameters are available.

Action	Result
Column name	Name of the new column. Source: Constant Data type: Date, Number, or Text Specification: Mandatory
Type	New column data type; Date, Number, or Text. Default value: Text; Specification: Mandatory
Create numeric enumeration	Fills a new column with ascending values. The values start at 1 or the value entered in the Value input box and increase by a value of 1 in each subsequent row. Specification: Optional If the option is enabled the Value box is disabled and any (default) value already entered or selected is deleted. Incoming connections for dynamic values are ignored.

Action	Result
Value	Initial value of the new column. Source: User input or constant and source table Data type: Depends on the data type of the source column. Specification: Optional

If a name of an existing column is specified as the column name, regardless of its data type, this has no effect on the result table, i.e., the original column values are retained. Existing columns are not overwritten by new columns with the same name. Multiple columns with the same name cannot exist in a table.

Error behavior

The following error behavior can occur.

Error situation	Behavior
Value does not match the column data type.	Error message.

Duplicate column

Behavior

Copies the specified columns from the data feed to new or existing columns of the same type.

It is possible to create multiple copies of a column but the target columns must have different names.

If the target column does not exist, it is created. If it does exist, it is replaced. All columns can be duplicated, regardless of type.

Parameter

The following parameters are available.

Action	Result
Source column	Name of the column to be duplicated.

Action	Result
	Source: Source table Data type: Date, Number, or Text Specification: Mandatory
Target column	Name of the new or existing column. Source: Constant Data type: Corresponds to source column. Specification: Mandatory

Delete column

Behavior

Deletes the specified columns from the data feed.

Parameter

The following parameters are available.

Action	Result
Column	Name of the column to be deleted. Source: Source table Data type: Date, Number, or Text Specification: Mandatory

Rename column

Behavior

Changes the names of the specified columns from the data feed. The data type of the column is retained.

Parameter

The following parameters are available.

Action	Result
Column	Name of the column to be deleted. Source: Source table

Action	Result
	Data type: Date, Number, or Text Specification: Mandatory
New name	New name of the renamed column. Source: Constant Data type: Corresponds to source column. Specification: Mandatory

Error behavior

The following error behavior can occur.

Error situation	Behavior
Column name and new name are identical.	Error message.
New column name already exists.	Error message.

Create Input Parameters

You can create input parameters to enter dynamic values that are passed into the data transformation step (e. g. for filtering) or passed to the data source itself.

By means of parameters sets you are enabled to create dynamic URLs for your XML source and RAQL queries. The parameters are used to dynamically pass context from dashboards to applications via URLs. You can create dynamic URLs and query parameters providing required context to open URLs, invoke web services, etc.

The functionality is available in the **Edit Data Source** dialog.

Procedure

1. Click the **Input Parameters** box.
2. Click the **Add Input Parameter** button.
3. Enter a **Name** for your parameter set.
4. Select the data type of the parameters, **Text**, **Number** or **Date**.

   The selection enables the dynamic entry of text, date or numerical values in data feed processing.
5. Enter an optional **Default value**.

The default value will be used if the input parameter is not filled by the dashboard context. For example, if the input parameter is filled by the selection of an other component and there is nothing selected.

6. Enter a **Preview value**.

   The preview value is used to calculate a data preview (debug run) in the data assignment dialog. See "data source assignment" on page 212.

7. Click the **Data Flow** box to display the data source overview.

The input parameter is displayed in the **Input Parameter** box. The defined input parameters are only available for the current selected component.

You can use the input parameters for the XML data source, RAQL queries and ARIS table of the current "data source assignment" on page 212. Available input parameters will be provided via an additional button by the data source operator.

If you use the input parameter in an XML data source URL, then you can specify, that the value of the parameter should be URL encoded. For an input parameter of **Date** or **Number** type you can additionally specify the format pattern.

In the BigMemory data source it's possible to specify the format pattern for an input parameter of **Number** and **Date** type.

To specify the parameter properties click on the inserted input parameter and select the option required. See also "data source assignment" on page 212.

The input parameters are also provided as filter elements by the dashboard component. See "Define filters for dashboard components " on page 276 for details.

Move components to front or back

You can move a component forward or backward in a dashboard relatively to other components. For example, you can display a chart in the background of the dashboard and place several components on top of it, i.e., in the foreground.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Select one or more inserted components in the dashboard. A corresponding pop-up menu is displayed.
3. Set **Layering**
   - Click the **Bring to Front** icon. Displays the component in front of one or more other components.
   - Click the **Bring Forward** icon. Brings the component one level forward.
   - Click the **Bring Backward**. Brings the component one level backward.

- Click the **Send Back**. Displays the component behind one or more other components.

The selected components are moved forward or backward in the dashboard.

Hide or display component header and border

You can hide the header and the outline of the component container.

The header and the outline are hidden for some components by default, e. a. Input field and Combobox. You can display them if necessary.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click on an inserted component in the dashboard. A corresponding pop-up menu is displayed.
3. Show the **Config** tab to set the display options.
4. Set the **Container** header and border.
   - Click the **Hide Header** icon. Hides the header incl. the title of the component and the content will be resized. Click the icon again to display the header.
   - Click the **Hide Border** icon. Hides the outline of the component container. Click the icon again to display the outline.

The header and outline of the component will be hidden or displayed.

Resize dashboard components

You can scale up or down the size of dashboard components.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click on an inserted component in the dashboard. The component will be displayed with a framing.
3. Resize the component by dragging the anchor point of the framing with your mouse.

The selected components are resized.

Copy and past components in dashboard

You can copy and cut components and past them in the same or in a other tab in the dashboard.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Select more inserted components in the dashboard. A corresponding pop-up menu is displayed.
3. Click the **Copy** icon in the pop-up menu to copy the selected components to the clipboard.
4. Click the **Past** icon in the pop-up menu to insert procedurethe copied components in the same tab of the dashboard.
5. Click the **Cut** icon in the pop-up menu to cut the selected components and copy them to the clipboard.

The selected components are copied to the clipboard.

You can insert copied components in any tab of the dashboard. Open a tab of the dashboard an press the **Strg**+**V**. Filter relations between copied components are retained.

Define filters for dashboard components

In the MashZone NextGen edit mode you can define relations between components by specifying filter conditions for selected components. The defined filters can be used in the dashboard view mode (See "Use interactive filter in your dashboards" on page 188).

Most components of the Dashboard support data filters. Particularly the **Drop-down box** and **Input field** provide you with a selection of values in a drop-down menu or a manual data input for filtering other dashboard components. (See "Insert Drop-down box in a dashboard" on page 208 and "Insert Input field in a dashboard" on page 209).

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click on an inserted component, e. g. **Line chart**, **Drop-down box** or **Grid**. The relevant properties dialog is displayed.
3. Click the **Filter** icon to edit the filter configuration. The selected component, the data source columns and the **Select filter component** dialog will be displayed.

   If applicable input parameters will be displayed as additional filter components. If input parameters have been defined, they are listed below the **Source columns** of a component. See"Create Input Parameters" on page 273 for details.
4. Drag a column that you want to be filter in the **Source Columns** box and drop the selected column in the **Drop her to define a new filter** field. The **Define filter condition** dialog will be displayed. All components that can be used as filter will be highlighted. The columns in the **Available Coordinates** box of each component can be used as filter condition values.

5. Define the filter condition.

   You can use a column or a constant value as filter condition.

   a. Select a condition in the drop-down menu, e.g. starts with.

   b. Drag a coordinate in the **Available Coordinates** box of a the component that you want to use as filer component. And drop the selected coordinate in the empty filter condition field.

   c. Alternatively, you can enter a constant value as filter condition.

   d. To consider the case sensitivity click the **Match case** / **Ignore case** icon. It's only available for coordinates or values of type *text*.

   e. If you want the filter conditions to be considered as fulfilled even if the selected values are empty click the **Empty comparevalues are accepted** icon.

   f. Click the **Add** icon to add a filter condition.

   g. Activate the **block values** option to block the rows which meets the condition.

6. Click **Define new filter** to add filters on other source columns.

7. Click **Save filter**.

Your filter conditions are saved.

Now you can "Use interactive filter in your dashboards" on page 188.

Delete components from dashboard

You can delete components from a dashboard.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Select one or more inserted components in the dashboard. A corresponding pop-up menu is displayed.
3. Click the **Delete** icon in the pop-up menu.
4. Click **Delete**.

The selected components are deleted from the dashboard.

Specify actions for dashboard components

You can assign actions to specific dashboard components (e.g., charts, traffic light, label, image). The actions set a data selection in other components, call other views of the dashboard or jump to a specific URL.

For an action, you can specify a data selection of a specific component on a dashboard view, e.g., column in a table. The component will then be displayed applying the data selected. If the data selected also represents filter values for another component, this component is filtered accordingly.

The actions are triggered by clicking a component, by a mouse over event or by selection change, depending on the component selected. **On mouse over** events are performed if you move your mouse pointer over a data point of a component, e.g. a coordinate of a Line chart. Whereas **On selection change** events are performed if you click on a data point in a chart or if you delete a data selection in a component. Depending on the component you can delete a selection by clicking in the background or selecting the **Delete selection** option in the ▼ **Menu** of the component.

For the Grid component additional trigger options can be available, depending on your Grid component settings. That are **On "item" click** and **On "item" traffic light click**. Whereby "item" is a placeholder for a column name. In the Grid component settings you can make column cells clickable for triggering actions. See "Assign data sources to Grid" on page 220 and "Configure thresholds of the Grid component " on page 222 for details.

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click an inserted chart component that supports actions. The relevant properties dialog is displayed.
3. Click the **Action** tab.
4. Select the action trigger event in the **Trigger** drop-down menu.

   The selection of actions is dependent on the selected component.
5. Activate the **Change Tab** action to call another dashboard view by action triggering and select the target tab in the drop-down menu.

   The action is only available if you have added one ore more views in the dashboard. See "Add views in a dashboard" on page 199.

   If you deactivate the option then the action will not be deleted but deactivated in the view mode.
6. Activate the **Set Selection** option to set a data selection by triggering an action. This action set a specific selection in one or several of the components on the same dashboard view.

   If you deactivate the option then the action will not be deleted but deactivated in the view mode.
7. Configure the **Set Selection** action.
   a. Click **Configure** to set the action configuration.

      Initially all **Available coordinates** of the target components are displayed.

b. Set the coordinates of the component you want to select by your action. For this drag the relevant coordinate and drop it in the field of the **Select selection component** area.

c. Set the values that should be selected. You can enter a constant value or you can assign the values of a coordinate of another component. Drag the relevant component coordinate and drop it in the **Selection** field of the previously set coordinate.

   Only the coordinates with the fitting values are provided.

d. Click **Add an additional selection** to define a further selection.

e. Click **Save action**.

8. Activate the **Call URL** option to jump to a specific URL by triggering an action.

   If you deactivate the option then the action will not be deleted but deactivated in the view mode.

   a. Click **Configure** to set the action configuration.

   b. Enter the target URL in **URL** input field of the **Enter Target URL** area.

      You have the option to create a dynamic URL by adding **Available Coordinates** to the URL. For this you can insert the coordinates from several components via drag and drop. A selected coordinate will placed on the current cursor position in the **URL** input field.

      In case of a coordinate of typ number or date, click the inserted coordinate and select a **Format pattern**.

      To ensure that a coordinate is URL encoded, click the inserted coordinate and activate the **Use URL encoding** option.

   c. In the **Target window** field you can enter a name of the window where the URL should be opened, or you can select a target attribute in the drop-down menu. Available attributes are **_blank** (new window), **_self** (self window), **_parent** (parent window) and **_top** (entire window).

   d. Click **Save action**.

Your action is specified for the selected component.

You can use the defined action in the dashboard view mode. See "Display dashboards in MashZone NextGen" on page 188.

Customize desktop

You can customize the dashboard desktop of MashZone NextGenDashboard.

Set display size

You can set the display size for the current dashboard in MashZone NextGenDashboard. Set the zoom factor to improve the clarity of a dashboard.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click **Options** in the Dashboard main menu.
   - Click **Zoom In** to increase the display size.
   - Click **Zoom Out** to decrease the display size.
   - Click **Set Zoom to 100%** to restore the default display size.

The display size of the current dashboard is adjusted accordingly.

Set grid lines

You can set the grid lines in the desktop background in MashZone NextGenDashboard. The grid lines help you to arrange the individual components more easily. The dashboard components are automatically aligned to the grid lines.

Procedure

1. "Create" on page 192 or "open" on page 192 a dashboard in MashZone NextGen Dashboard.
2. Click **Options** in the Dashboard main menu.
   - Click **Increase Grid Space**.
   - Click **Reset Grid** to to restore the default grid space.
   - Click **Decrease Grid Space**.

The grid lines of the current dashboard are adjusted accordingly.

5 Mashups in MashZone NextGen Wires

- Wires Features 282
- Creating Mashups in Wires 284
- Manage Mashups in Wires 386
- Edit Mashup XPath Expressions in Advanced Mode 391
- Customizing Wires 392

MashZone NextGen Wires gives you an easy, graphic way to find information within your organization or from outside sources and combine or transform it in a mashup. Your information sources include MashZone NextGen mashables and mashups, public resources accessible on the Internet or information from SharePoint, if the MashZone NextGen Add-On for SharePoint is installed with MashZone NextGen.

Wires lets you drag information sources as simple, graphic blocks, add action blocks to transform information and connect blocks with "wires" to create a mashup for your use or for other users. You quickly create basic mashups.

Mashups combine and transform the information, but don't let you see the information or share it easily. Apps let you view results from mashable information sources or mashups in different formats and easily add these views to MashZone NextGen workspaces, wiki pages, portal pages, SharePoint or any web page.

So how do you get started? See "Wires Features" on page 282 for an introduction to the design features in Wires. For common questions, try the "MashZone NextGen concepts" on page 66.

Once you are ready to create your first mashup, start with Creating Mashups in Wires.

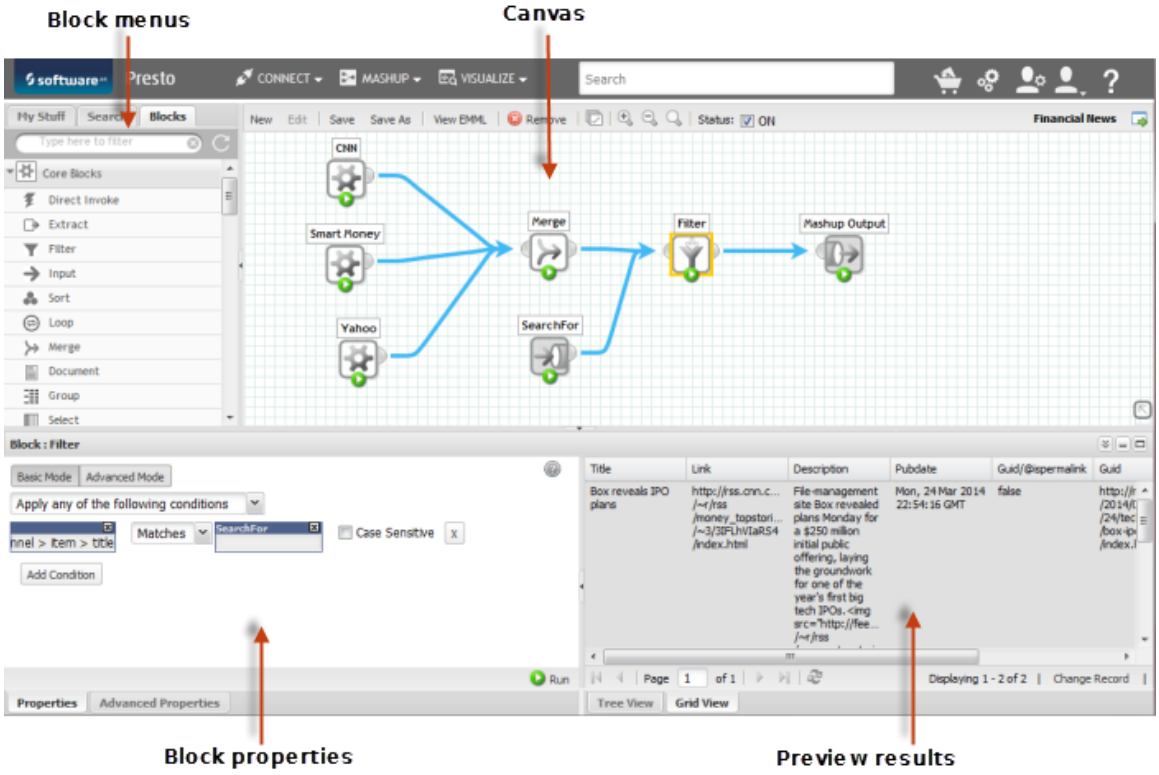
Once you have a mashup, you must add views to the mashups to let you or other users see mashup results in useful ways. See "Views for Mashups and Mashables" on page 895 for links to more information.

When the mashup has views, you can then create one or several apps to begin using it in web pages or a wiki. See "Apps and Workspaces" on page 1091 for links to more information on using mashups in apps.

For more robust or complex mashup capabilities, you create mashups using EMML and the Mashup Editor. See "Mashups in EMML" on page 441 for more information.

Wires Features

You open Wires from **Mashups >** Wires in the MashZone NextGen Hub main menu. The basic features that you use in Wires include:



Note: Some features in Wires may not be accessible to you based on your MashZone NextGen permissions.

- **Design Canvas**: is where you create or edit mashups. You can drag or drop *blocks* from the Block Menus into the canvas and then connect them to define how the mashup should work.

  The canvas also contains the Wires toolbar with basic editing actions. Some features that may be new to you include:

  - **Status: On** or **Status: Off** options allow you to change the status of the mashup.

    Mashups that are on are considered "published". They are accessible to other users and can be used in other mashups or in apps. You must turn mashups off before you edit them.

    Mashup status is also automatically updated when you save the mashup. The status is turned on if all blocks are fully configured and results are connected to the mashup output or turned off if not.

  - **Select All** to select all blocks on the canvas so that you can move them together. You can also use `Ctrl` + `A`.

  - **View EMML** to see the raw XML statements for the mashup. You can also copy these commands to the clipboard. You can also use `Ctrl` + `U`.

- **View Mashup Information** to see basic information about mashups that have been saved.
- **Open Details** to go directly to the MashZone NextGen Hub page for this mashup where you can add views, take snapshots or create apps based on this mashup.
- **Pan** to pan the visible portion of the canvas over large mashups.

You can also use `Ctrl` + `S` to save the mashup and `Ctrl` + `N` to start a new mashup.

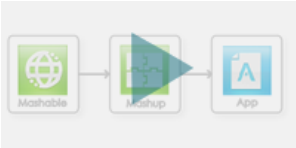
- **Block Menus**: contains tabs with menus and search to help you find mashups, mashables and other *blocks* that you can use in a mashup.
  - Use **My Stuff** to work with the mashups and mashables that you have created or registered or other mashables and mashups that are your favorites.
  - Use **Search** to search for any mashable or mashup by name, description or tag.
  - Use **Blocks** to find the actions or other blocks that help you create the mashup.

  You can also:
  - Use the filter to shorten menus and find specific blocks more quickly.
  - **Open** a mashup for editing. You can only edit mashups that you have created, unless you are a MashZone NextGen administrator.
  - **Information** to view basic information for a mashup or mashable.
- **Block Properties**: shows you the input or properties that you can set for the block that is currently selected in the design canvas.
- **Preview Results**: allows you to view the structure and data in the result of any block in the mashup, including the final mashup result.

Flags

- *Inactive* mashables or mashups
- *Loading*
- @ The field is an XML attribute

Creating Mashups in Wires



You create mashups in Wires using blocks in a canvas that represent the data sources and actions to apply to data. You connect blocks with wires to determine how mashup processing should proceed.

1. Select **Mashups >** Wires in the MashZone NextGen Hub main menu.

   Wires opens with a canvas containing an **Output** block. This required block represents the results of the mashup.

2. Drag blocks from the menus into the design canvas for the information sources and actions to create the mashup result. See:

   - "Add a Mashup or Mashable as an Information Source" on page 286, "Add a Mashup or Mashable in a Loop" on page 295, "Add a Web Information Source with DirectInvoke" on page 288 or "Add SharePoint Information Sources" on page 290 for instructions on adding information sources.
   - Add Actions or Other Blocks for links to all the built-in actions to select, combine or transform information.
   - Select Fields or Paths for Block Properties with the Path Selector to provide dynamic values, if needed.
   - Preview Results at any step in the mashup or for the mashup itself.

     It is a good practice to run each block to preview results as you build your mashup. You can also Stop Running a Block if problems occur with preview.

   You can also change block labels to make your mashup more readable. Simply double-click in the label and make your change.

3. Connect Mashup Blocks to determine how information is processed.
4. Once you have the result you want, connect that block to the **Output** block. If needed, you can Quickly Set All Inputs for the Mashup, Change the Mashup Result Character Encoding or Add HTTP Headers to a Mashup from **Output**.
5. Click **Save** and:

   a. Enter a **Name** for the mashup.

      MashZone NextGen uses the mashup name to assign a unique identifier to the mashup. Mashup names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

   b. Enter or select optional information including:

      - A description of the purpose of this mashup.
      - The name of the information provider to assign to this mashup.
      - Tags for this mashup.
      - A category for this mashup.

c. Click **OK**.

When you save a mashup, Wires automatically turns the mashup status to ON unless either:

- One or more blocks are not fully configured.
- Results from the final action are not connected to the Output block.

You can use the **On|Off** button in Wires to manually manage the mashup's status.

When the mashup status is OFF, you can continue editing and refining the mashup. But no one can add views, take snapshots, create apps or use the your mashup in a workspace until the status is ON, also sometimes called publishing the mashup.

Once the mashup is on, you or MashZone NextGen administrators *must* do the following to allow other users to work with it or allow it to be used in apps or workspaces:

- Add views for this mashup in the mashup's artifact page. See "Open the Mashup Artifact Page to Create Apps, Add Views or Take Snapshots" on page 390 for more information.
- Assign permissions to determine who can use this mashup. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for more information.

Add a Mashup or Mashable as an Information Source

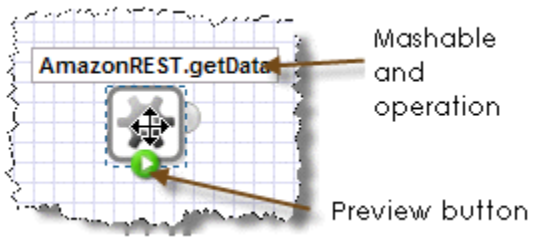
1. Find and select the mashup or mashable you want in the Block Menus.

   See "Quickly Find Mashables, Mashups or Other Blocks" on page 387 for search tips.

   This opens a list of operations for this mashable or mashup in the Block Menu.

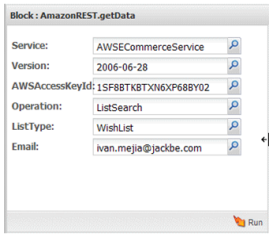
2. Select the operation you want to use and click to add it to the design canvas or simply drag the operation to the position you want.

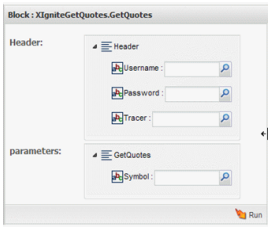
   The block appears in the canvas with either the **Preview** button or the **Inactive** flag.



Note: You can use inactive mashables or mashups in your mashup, but you cannot preview results for these blocks. You also cannot use the mashup until those services or mashups are activated.

3. Set any of the properties you need in the Block Properties pane.





Properties may be a simple list or they can be organized in a hierarchy. Properties may also have default values.

Properties may accept values directly or open the **Path Selector** list to allow you to supply the value from input fields or other blocks.

a. If needed, expand the hierarchy of properties.

b. Click in any property field and:

- Enter a value.
- If the **Path Selector** list opens, add an input field, select an existing input field or select a path to a field from another block to supply the value for the property dynamically. You can also click to open the **Path Selector** list.

  See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for instructions.
- Enter a MashZone NextGen global or user attribute to supply the value for a property. See "Use MashZone NextGen Global or User Attributes in Block Properties" on page 374 for instructions.

c. Click **Run** to preview the results.

It is a good practice to preview the results for this information source. Wires typically shows the results in the Tree view, to show both the structure and data of the result. You can also view just the data in the Grid view:

Title	Link	Description	Pubdate
Medicare doctor: Here's what I make	http://rss.cnn.com/~r/rss/money_topstories/~3/nhZ8O7U4VCg/index.htm	When you think of low-paying jobs, doctor doesn't usually come to mind.	Thu, 04 Mar 2010 18:47:44 GMT
World's Most Admired Companies	http://rss.cnn.com/~r/rss/money_topstories/~3/KEedhlux0yI/670.html	Which companies have the best reputations? Apple tops the list for the third year in a row. See who	Thu, 04 Mar 2010 18:45:43 GMT

Page 1 of 1 | Displaying rows 1 - 20 of 20

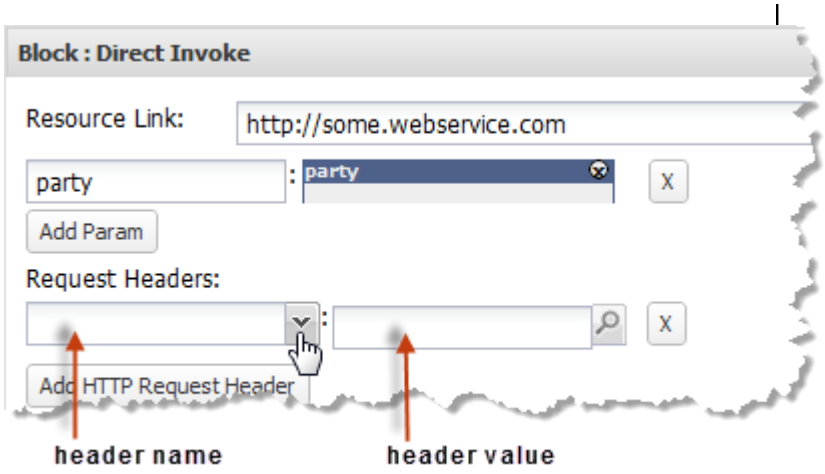
Note: Some mashables and mashups return a large volume of results. Wires displays only the first 50 results. If you wish to see more results, your MashZone NextGen administrator can change this setting. Be aware that large numbers of results can cause your browser to hang and stop responding.

Add a Web Information Source with DirectInvoke

Use the DirectInvoke action to add information directly from web sites or web services available on the Internet.

ResourceLink	This is the URL to the web site or web service to use. Most web sites and some web services use the HTTP method `GET` so the URL also defines the specific request to send to the web site or web service in one or more parameters. Some web services use the `POST` method instead of `GET`, or support the `PUT` or `DELETE` methods. You can either: ■ Enter a URL. ■ Click to open the **Path Selector** list to use a dynamic URL. Add an input field, select an existing input field or select a path from the results of any blocks in this mashup to supply the URL. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for details.
Add Param	If the method is `GET` or `POST`, click this button to add a parameter to this URL. You can add any number of parameters.
Parameters	Enter the name of the parameter in the left field. **Note:** Parameters are *only* valid when **Method** is `GET` or when web services use `POST` with *name/value pairs*. For more information, see "<directinvoke>" on page 478 in EMML. For the parameter value, either: ■ Enter a value directly. ■ Click to open the **Path Selector** list to make the parameter value dynamic. Add an input field, select an existing input field or select a path from the results of any blocks in this mashup to supply the parameter value. See

	"Select Fields or Paths for Block Properties with the Path Selector" on page 372 for details.
Method	Choose the HTTP method to use to send this request to the web site or web services: ■ `GET` = the default and most common method, especially for web sites. With `GET`, all information for the request, including parameters, is defined in the URL. Parameters have only simple values. ■ `POST` = choose this method for web services or web sites that need larger or more complicated requests or requests in specific formats such as XML. ■ If the request uses *name/value* pairs, add them as parameters. ■ Otherwise, define the request content in **Post Data**. You must also define the format of the content as a *content-type* header. For more information, see"<directinvoke>" on page 478 ■ `PUT` = this method is used by some web services to add or update a specific resource. For example, a web service for airline reservations might use the `PUT` method with a URL to allow you to make a flight reservation. You define the content for this request in **Post Data**. The content and format to send depends on the requirements for this web service. You may also need to add HTTP headers for this request. For more information, see "<directinvoke>" on page 478. ■ `DELETE` = this method is used by some web services to delete a specific resource. For example, a web service for airline reservations might use the `DELETE` method with a URL to allow you to delete a flight reservation. You may need to define content for this request in **Post Data** or add HTTP headers for this request, based on what the web service requires. For more information, see "<directinvoke>" on page 478 in EMML
Add HTTP Request Header	Click this button to add one or more HTTP header to this request to this web site or web service: ■ In the left header field, either select the standard HTTP header that you want or directly enter the name of a vendor-specific HTTP header.



- Enter the value for this header in the right field. Or click to open the **Path Selector** list to set this value dynamically. Add an input field, select an existing input field or select a path from the results of any blocks in this mashup to supply this value. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for details.

Post Data	Defines the content of the request to send when **Method** is `POST`, `PUT` or `DELETE` and the web site or web service does *not* expect *name/value pairs*. You can either: ■ Enter the content directly as XML or in whatever format is required. If the content is XML, it must be *well formed* (completely enclosed within one root element). ■ Click to open the **Path Selector** list to find a document that defines the content or to make the content dynamic. Add an input field, select an existing input field or select a path from the results of any blocks in this mashup to supply the content. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for details.

See also "Define Error Handling" on page 382 and "Use a Security Profile with DirectInvoke" on page 382 for information on advanced properties for DirectInvoke.

Add SharePoint Information Sources

If the MashZone NextGen Add-On for SharePoint is installed and configured by your MashZone NextGen administrator, you can use certain types of information in SharePoint as information sources for mashups in Wires:

- SharePoint Lists that are registered as mashable information sources (from MashZone NextGen Hub and from SharePoint) can be used in mashups just like any other mashable. Lists registered as mashables can also be used to create basic apps or workspace apps.
- You can add SharePoint Lists, or specific views for a SharePoint List as an information source in mashups using the *SharePoint List* block. See "Add a SharePoint List or List View to a Mashup" on page 291 for instructions.
- You can add the results of a SharePoint search as an information source in mashups using the *SharePoint Search* or *SharePoint FAST Search* action blocks. See "Add Results from a SharePoint Search or FAST Search to a Mashup" on page 292 for instructions.

Add a SharePoint List or List View to a Mashup

Use the *SharePoint List* block to add a list or one view of a list in SharePoint as an information source for the mashup. Add the block to the canvas and set these properties:

Connection	Select the connection for the SharePoint server with the site and list that you want to use. **Note:** If no SharePoint connections have been configured in MashZone NextGen, this property shows a message and you cannot use this block until a connection to SharePoint has been configured by a MashZone NextGen administrator. If the SharePoint server you need to work with is not in this list, contact your MashZone NextGen administrator.
User Credentials	This section is collapsed by default. It defines the credentials (user account and permissions) used to connect to this SharePoint server and sites. Use the and buttons to expand or collapse this section. The permissions for the credentials you use determine which SharePoint sites and lists you can use in this block. Based on configuration, this may default to the credentials defined for the connection you have selected, also known as the *service account*. Or you may be required to supply your credentials for this SharePoint server. **Note:** Some SharePoint server connections may disable the use of user credentials based on configuration. In this case, this section only displays credential information.

	To use your own SharePoint credentials to connect to this server, change or enter this account information and click **Save**. **Important:** Updating these credentials creates or updates *your* MashZone NextGen credentials for this SharePoint server (which are saved as MashZone NextGen User Attributes). This can affect other mashups or mashables that work with this SharePoint server if they use your credentials rather than the service account. To remove your credentials from this block, click **Remove**. Depending on configuration for this server, the service account credentials may default. This also *removes* your MashZone NextGen User Attributes for this SharePoint server.
Site Name	Once you have the connection and user credentials set, choose the SharePoint site with the list you want to use.
List Name	Choose the SharePoint List you want to use as an information source.
View Name	If desired, choose the specific view that you want to use for this list. Views determine the specific columns that are visible from the SharePoint List.
CAML Query	Optionally, enter a query, using the Collaborative Application Markup Language (CAML) query syntax, to further refine the List items and data you want to retrieve.

Add Results from a SharePoint Search or FAST Search to a Mashup

Use the *SharePoint Search* block to use the results of a simple search in SharePoint as an information source. Complete the "Server, Credentials and Site Properties" on page 293 and "Simple Search Properties" on page 294 for this block.

Use the *SharePoint FAST Search* block to use the results of an advanced search in SharePoint as an information source. FAST advanced searches use the FAST Query Language (FQL) to allow advanced searching such as wildcard matches, complex queries such as both terms present (AND/OR) or filtering results to specific properties.

Note: The FAST search feature is *not* available in SharePoint 2007.

For advanced searches, complete the Server, Credentials and Site Properties and FAST Search Properties for this block.

Server, Credentials and Site Properties

Connection	Select the connection for the SharePoint server with the site you want to search. **Note:** If no SharePoint connections have been configured in MashZone NextGen, this property shows a message and you cannot use this block until a connection to SharePoint has been configured by a MashZone NextGen administrator. If the SharePoint server you need to work with is not in this list, contact your MashZone NextGen administrator.
User Credentials	This section is collapsed by default. It defines the credentials (user account and permissions) used to connect to this SharePoint server and sites. Use the and buttons to expand or collapse this section. The permissions for the credentials you use determine which SharePoint sites and lists you can use in this block. Based on configuration, this may default to the credentials defined for the connection you have selected, also known as the *service account*. Or you may be required to supply your credentials for this SharePoint server. **Note:** Some SharePoint server connections may disable the use of user credentials based on configuration. In this case, this section only displays credential information. To use your own SharePoint credentials to connect to this server, change or enter this account information and click **Save**. **Important:** Updating these credentials creates or updates *your* MashZone NextGen credentials for this SharePoint server (which are saved as MashZone NextGen User Attributes). This can affect other mashups or mashables that work with this SharePoint server if they use your credentials rather than the service account. To remove your credentials from this block, click **Remove**. Depending on configuration for this server, the service

	account credentials may default. This also *removes* your MashZone NextGen User Attributes for this SharePoint server.
Site Name	Once you have the connection and user credentials set, choose the SharePoint site that you want to search.

Simple Search Properties

Search Query	Enter a word or phrase to search for in this SharePoint site.
Start At	If desired, change the index number for the first result to return. This defaults to `1`.
Count	If desired, change the total number of results to return. This defaults to `0` which returns all results.

FAST Search Properties

Query Type	This defaults to `Keyword` which searches for simple text. Change this to `FQL` to use an advanced query expression to define what to search for.
Search Query	If the query type is `Keyword`, enter a word or phrase to search for in this SharePoint site. If the query type is `FQL`, enter a query in FQL for what to search for in this SharePoint site.
Spell Check	This option is set by default. It checks spelling within the keywords of the query and potentially returns other suggestions.
Resubmit if zero results	If desired, set this option to allow the search to turn *off* spelling checks and turn on *stemming* to attempt to find other results if an exact match does not find any results.

Add a Mashup or Mashable in a Loop

Use the **Loop** action block to add mashup or mashable results in a repeating loop in your mashup.

The input to this block must have a set of repeating items that control looping. Each repeating item in the input runs the mashup or mashable in one loop and may also be used to provide inputs to the mashup or mashable. The loop ends when the last repeating item is processed.

The final results for the block, in most cases, consist of all the results from each loop. You can, however, simply add the results from each loop to the corresponding repeating item in the input to the block.

1. Add **Loop** to the canvas and connect a document-type result *with at least one repeating node* as the input to this block.
2. If needed, set the **Select Repeating Element** property to point to the repeating node to define this loop.

   If there is only one repeating node from the input, this property defaults to that. You can also click to open the **Path Selector** list and find the appropriate repeating node. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for instructions.
3. If needed, change the **Result Method**.
   - `Append` is the default method for adding in the mashup or mashable results from each loop. Each loop result appears as a separate 'item' in the accumulated results for this block.
   - `Merge` combines the results of each loop into a single 'item' rather than separate items. Merged results can be more uniform, but also obscure which portions of the accumulated result came from a specific loop.
   - `Annotate Source` takes the results of each loop and adds that as a child to the corresponding repeating item in the input to the **Loop**block. Thus the final result of the block is the input document annotated with the loop results.
4. For appended results, you change the name of the root node wrapping all loop results in the **Results Name** property.

   This defaults to `result`.

5. If the results of your mashup or mashable information source can have different structures, enter the name of the node within the results where they should be merged in **Merge On**.

   Note: If the output of the mashup or mashable does not vary, simply leave this field blank.

6. Add the mashup or mashable to invoke in the **Operation to invoke for each item** canvas:

   a. Find and select the mashup or mashable you want in the Block Menus.

      See "Quickly Find Mashables, Mashups or Other Blocks" on page 387 for search tips.

      This opens a list of operations for this mashable or mashup in the Block Menu.

   b. Drag the operation into the canvas in the **Loop** block properties.

   The canvas disappears and information on the mashup/mashable operation displays. If this mashup or mashable has input parameters, new properties open for these parameters.

   You can also use the **Clear** button to remove this mashup or mashable.

7. Complete the input parameters, if any, for the mashup or mashable you have added to the loop.

   In many cases, you provide input parameters for the mashup or mashable from values in the repeating item that defines the loop, although this is not required. Enter values or use the **Path Selector** list to complete input parameters just as you normally would.

Preview Results

You can preview the results for a mashup or for any active block in a mashup. Simply click **Run** on any block to see results for that block. **Run** is not available on blocks for inactive mashable information sources or mashups.

Tip: It is a good practice to run each block once you have set or changed properties or inputs. This also makes it easier to work with block properties.

Wires displays the results in the Tree view, which shows the structure of the result. You can also use the Grid view to just see data.

Tip: Network outages or other issues can cause preview to take too long. You can also Stop Running a Block to cancel the preview.

About Results in Wires

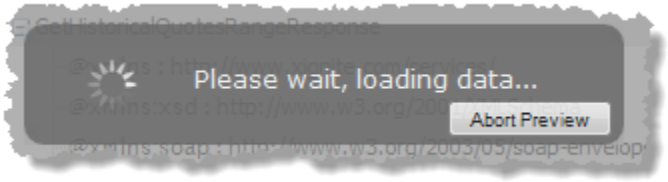
Some mashables and mashups return a large volume of results. Wires displays only the first 50 results to help ensure good performance when you preview blocks.

If you wish to see more results, your MashZone NextGen administrator can change this setting. Be aware that large numbers of results can cause your browser to hang and stop responding.

Stop Running a Block

Problems can occur when you run blocks because of network outages or other issues. When you click **Run** on a block, or in Block Properties, to preview the mashup results at that step, you can cancel the preview if it takes too long.

Use the **Abort Preview** button in the loading message that appears when you run a block to stop waiting for a result for this block:



Add Actions or Other Blocks

Blocks that are not MashZone NextGen mashables or mashups appear in the Blocks Tab. Most of these additional blocks are *actions* that you can use to change results or provide dynamic inputs to the mashup. Some additional blocks also allow you to use information sources from sources other than MashZone NextGen, such as web sites, databases or SharePoint.

In most cases to use an action block, you add the block to the canvas, connect one or more other blocks as input to the action and then set additional properties for the action block.

Wires has a set of built-in actions and other blocks including:

Core	■ **Direct Invoke**: to add information sources from the Internet or other URLs. See "Add a Web Information Source with DirectInvoke" on page 288. ■ *Document*: to build a well-formed document as input to a single property for another block. Common examples are for complex input parameters or headers for mashables or other mashups. See "Create a Document for Input" on page 311.

	■ *Extract*: to extract one field, several fields or one or more repeating items from a result. The extracted results can be used as input to another block or as the final result for the mashup. See "Extract Partial Results" on page 318. ■ *Filter*: to select specific items in a result based on one or more conditions. See "Filter Results" on page 319 and "Filtering Techniques" on page 322. ■ **Group**: groups repeating items based on the unique values of one or more fields. This can also optionally calculate basic statistics for groups. See "Group Results" on page 323. See also "Aggregate Statistics" on page 300. ■ *Input*: to provide dynamic input fields for other blocks. See "Add Input Parameters" on page 327 or "Add Input Fields Directly to Block Properties" on page 303. ■ **Join**: to combine the repeating items from two blocks based on a relationship. This relationship in the *join condition*. See "Join Results" on page 330. ■ *Loop*: to add a mashable or mashup and run it in a loop for multiple results. See "Add a Mashup or Mashable in a Loop" on page 295. ■ *Merge*: to combine several results that all have the same structure. See "Merge Results" on page 337. ■ *Parallel*: to run two or more mashables or mashups at the same time (in parallel), generally to improve performance. For more information, see "Run Several Blocks Simultaneously" on page 340. ■ *RAQL*: to work with and analyze large datasets using MashZone NextGen Analytics and the Real-Time Analytics Query Language (RAQL). See "Analyze Datasets with RAQL Queries" on page 343. ■ **Select**: to select only specific fields in every repeating item in a result. See "Select Fields" on page 345. ■ *Sort*: to sort repeating items in a result based on a field. See "Sort Results" on page 352.
Data Transformation	■ *CSVGenerator*: to convert results to a CSV (comma-separated values) format used in spreadsheets and some databases. See "Format Results as CSV" on page 307. ■ *DataDecorator*: to convert, format or transform just the data in results from another block. This action block allows you to apply

	one or several functions to data. See "Decorate or Transform Data" on page 307. ■ **Mapper**: to change the structure, organization or data of results to match a known structure. This *maps* the two structures. See "Map Results to Known Structures" on page 334. ■ **Transformer**: to change the structure, organization or data of results to a structure you define. See "Transform Results" on page 354.
Database	**Database**: to run one SQL statement for any datasource that your MashZone NextGen administrator has configured. You can use this block to query a database, run a stored procedure or insert, update or delete records. See "Run a SQL Statement" on page 348.
Math	■ **Average**: to calculate the average value of one field in a set of repeating items. See "Average Results" on page 304. ■ **Counter**: to count the number of items in a set of repeating items. See "Count Results" on page 304.
SharePoint	Blocks for SharePoint*only* function properly if your MashZone NextGen license includes the MashZone NextGen Add-On for SharePoint and it has been configured by your MashZone NextGen administrator. ■ You can get information for a mashup from SharePoint Lists or SharePoint search results using the following blocks: ■ **SP List**: to add a SharePoint List, or a specific view for a SharePoint List as an information source in a mashup. ■ **SP Search**: to add the results of a simple SharePoint search directly as an information source in a mashup. Searches allow you to filter results from a SharePoint service, such as an RSS feed. ■ **SP FAST Search**: to add the results of an advanced SharePoint search using the FQL query language as an information source in a mashup. See "Add SharePoint Information Sources" on page 290 for instructions. ■ **External Content Type Adapter**: allows a mashup to act as an external data source for SharePoint Lists. Adding this block to a mashup is required to allow SharePoint users to define the mashup as an *external content type*.

	Note: This block makes mashup compatible with the Microsoft Business Connectivity Service (BCS). Lists based on external content types allow SharePoint to retrieve dynamic information from 'external sources' such as mashups, and thus many MashZone NextGen mashables. ■ **Merge** SharePoint **Lists**: allows you to merge the information from two or more SharePoint Lists. ■ **Add Item to a** SharePoint **List**: allows you to add an item to an existing SharePoint List and get either the full results for the List including the new item or detailed results for the item just added. .
Utility	■ **Aggregate**:performs basic statistics for repeating items that are grouped based on one or more key fields. See "Aggregate Statistics" on page 300. See also "Group Results" on page 323. ■ **DateBuilder**: to select a date and apply one of several date formats. See "Build Dates" on page 309. ■ **NamespaceStripper**: to remove all namespaces from a result. See "Strip Namespaces" on page 340. ■ **StringBuilder**: to build a string of text from extracted values, input fields or other dynamic data. See "Build Strings" on page 351. ■ **URLBuilder**: to build a URL from extracted values, input fields or other dynamic data, typically for use with Direct Invoke. See "Build URLs" on page 369.

MashZone NextGen administrators or developers can also add custom blocks to Wires. Contact your MashZone NextGen Administrator for information on working with custom action blocks. See "Adding Custom Blocks to MashZone NextGen Wires Using Macros" on page 392 for more information.

Aggregate Statistics

Use the **Aggregate** block to perform simple calculations on groups of repeating items. Each group is based on the unique values of one field. Optionally, you can filter items for a group or calculate simple statistics for the group. You can define multiple levels of groups.

Note: This block is very similar to the **Group** block. The primary difference is that Group includes data for all the repeating items within its groups. Aggregate *only* output the unique values for each group and the statistics defined for that group.

1. Connect a document-type result as input to the **Aggregate** block.
2. Use the Path Selector in **Select Repeating Element** to identify the repeating items to be grouped.

   See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
3. Define the group in **Group Aggregate**:
   a. Use the Path Selector in **Group By Key** to find the field within these repeating items to define the group for this level. Each unique value in this field will define one group.
   b. Optionally, enter a name for the node that should wrap the statistics for this group in **Group Name**.

      Note: Names must be valid XML names, starting with a letter and containing only letters, numbers, underscores (_), dashes (-) or periods (.).

      If you omit this property, the root node wrapping all groups defaults to `output`. For lower level groups, omitting this property omits the wrapping node.
   c. If needed, "Set Conditions to Filter Items to Include in Groups" on page 302.
4. Then complete the equation for at least one aggregate statistic:
   a. In the left field, enter a name for the result node to contain this calculation.

      Note: Names must be valid XML names, starting with a letter and containing only letters, numbers, underscores (_), dashes (-) or periods (.).
   b. Choose the function for this calculation:
      - `count`
      - `sum`
      - `avg`
      - `min`
      - `max`
   c. Use the Path Selector to find the field within these repeating items to use for this calculation. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

      Note: Except for the `count` function, this field *must* have numeric data.
5. If needed, click **Add an Aggregate** to add more calculations for this group.
6. If you need statistics for additional levels of grouping, click **Add Nested Aggregate**.

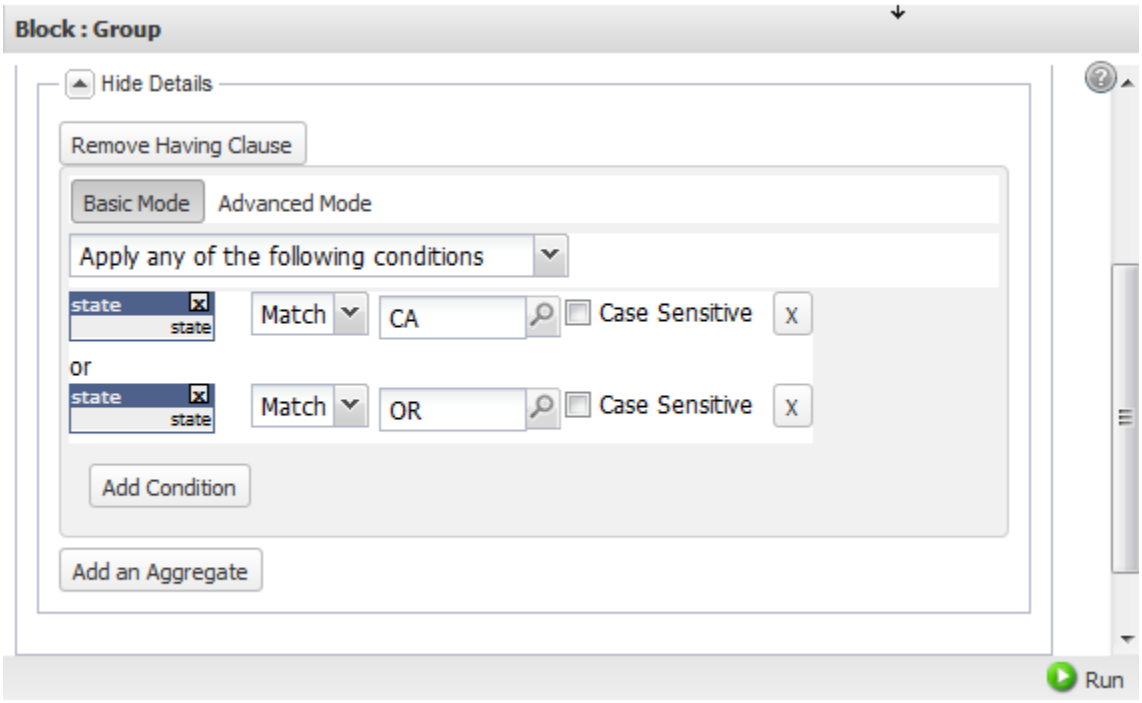
This opens a new **Group Aggregate** section. Complete this section as needed (see previous step) and continue adding **Group Aggregate** sections to add additional groups as needed.

See "Example 5. Aggregate Configuration Example" on page 303 for an example of group configuration and the output.

Set Conditions to Filter Items to Include in Groups

To filter which repeating items are included in a group, you add a "having" clause..

1. Click **Add Having Clause** for the group you want to filter. This adds fields defining the filter condition to use to include items in the group.



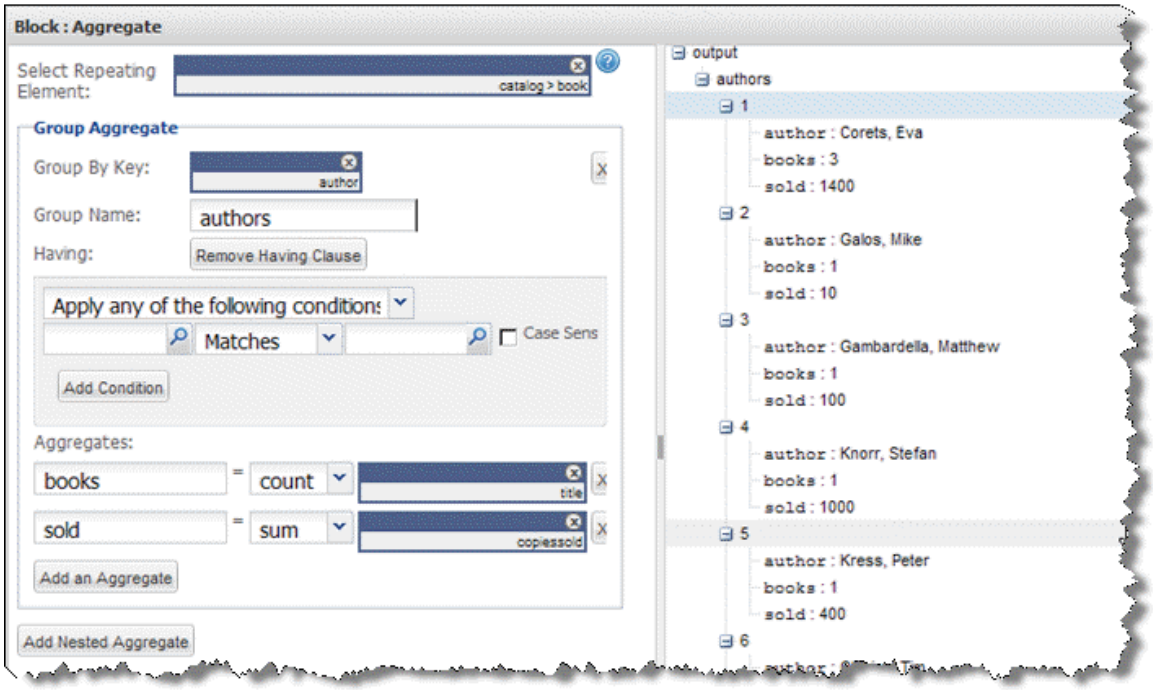
2. In the left field, use the Path Selector to find the field to compare for this filter. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
3. Choose the comparison operator.

   Use the basic math operators (=, <, <=, >, >=) for numeric fields. Use `Matches`, `Does Not Match`, `Contains` or `Does NotContain` operator for partial matches in text fields.
4. Enter a value to compare this field to or use the Path Selector to find the field for this comparison. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Note: MashZone NextGen developers who know XPath well can use the *Advanced Mode* button to customize the XPath expressions for conditions. Use the advance mode with care, however, as it disables the display of conditions in basic mode. If you return the condition to basic mode, your customized XPath expression for the condition is lost.

See "Edit Mashup XPath Expressions in Advanced Mode" on page 391 for more information.

Aggregate Configuration Example



Add Input Fields Directly to Block Properties

You can automatically create an input field for many block properties including parameters for information sources, values for some action properties or values used in conditions.

Note: Wires automatically assigns simple numbered names to input fields, but you *cannot* change the name. Change the label instead.

To automatically generate an input field, click in a block property field or click to open the **Path Selector** list. Select **Add as input** from the list.

This adds an **Input** block and automatically connects it to the block property. Set type or default value information for this input field as needed.

Average Results

The **Average** action calculates the average value of a field, with numeric values, in a set of repeating items. Connect the results with this numeric field as an input to this action.

Note: The result from this block is a number, rather than a well-formed document.

Average On	Click in this field to select the field within repeating items to calculate an average value for. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Count Results

The **Counter** action calculates the number of items in a set of repeating items. You can also set one or more conditions that items must match in order to be counted.

You must connect document-type results as input to this action.

Note: The result for this action is a simple number rather than a well-formed document.

Select item to count	Click in this field to select the repeating items that you want to count.See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Add Conditions / Remove All Conditions	To filter the set of repeating items that should be counted, click **Add Conditions** and complete one or more conditions. To remove all conditions, click **Remove All Conditions**.
apply condition	If you add multiple conditions, use this option to determine whether all conditions or only one must be met for an item to be counted.
left condition field	Click here to select the field in each repeating item that you want to use to filter the items to count. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
comparison function	If needed, change the comparison function for the filter condition:

	■ *Contains* = checks to see that the text characters you are filtering for are present somewhere in the field you have chosen to filter by. The comparison is not case sensitive, unless you set the Case Sensitive option. ■ *Does not contain* = checks to see that the text characters you are filtering for are *not* present somewhere in the field you have chosen to filter by. The comparison is not case sensitive, unless you set the Case Sensitive option. ■ *Matches* = checks to see that the characters you are filtering for are present somewhere in the field you have chosen to filter. ■ *Does not match* = checks to see that the characters you are filtering for are *not* present somewhere in the field you have chosen to filter. ■ *= (equals), != (does not equal), > (is greater than), < ()is less than, >= (is greater than or equals) or <= ()is less than or equals* = compares the entire value of the field you select with the value you set for the filter condition using standard numeric comparisons.
right condition field	Set the value to check for in the right side of the condition. You can: ■ Enter a value, word or phrase directly in the right side of the filter condition. ■ Use an input field to provide the value dynamically. See "Selecting Existing Input Blocks for Block Properties" on page 306 or "Add Input Fields Directly to Block Properties" on page 303 for instructions. ■ Use a MashZone NextGen global or user parameter to provide the value. See "Use MashZone NextGen Global or User Attributes in Block Properties" on page 374 for instructions.
Case Sensitive	Set this option if the condition should be case sensitive.
Add Condition	Add another condition.

Note: MashZone NextGen developers who know XPath well can use the *Advanced Mode* button to customize the XPath expressions for conditions. Use the advance mode with care, however, as it disables the display of conditions in basic mode. If you return the condition to basic mode, your customized XPath expression for the condition is lost.

See "Edit Mashup XPath Expressions in Advanced Mode" on page 391 for more information.

Use Input Fields in Block Properties

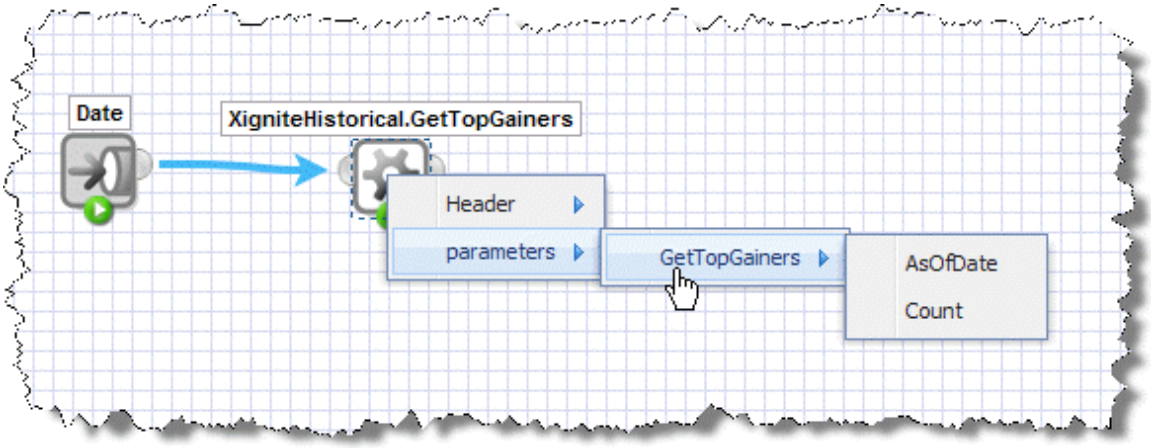
You can use Input blocks to define input fields for the mashup to provide dynamic values for properties in other blocks. There are several ways to connect an Input block to a block property:

- By drawing connections in the canvas. See "Connecting Input Blocks to Block Properties in the Wires Canvas" on page 306 for instructions.
- By selecting an existing input field in the Path Selector list. See "Selecting Existing Input Blocks for Block Properties" on page 306 for instructions.

Connecting Input Blocks to Block Properties in the Wires Canvas

For information source properties, you can add Input blocks and then simply connect them to the mashup or mashable blocks in the canvas..

1. Add an Input block to the canvas.
2. For mashup or mashable blocks, draw a connection from → **Input** to the mashable or mashup block.
3. If the mashable or mashup has several properties, choose the property to assign this input field to.



As this example shows, properties may also be in a 'header' for the mashable information source or mashup.

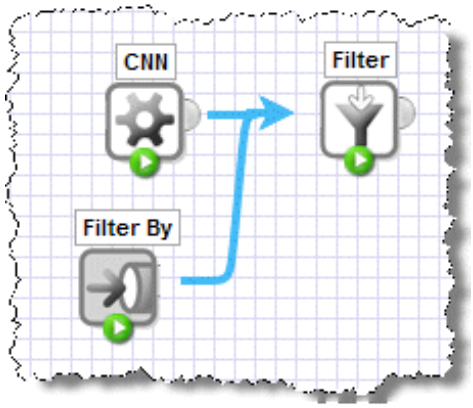
Selecting Existing Input Blocks for Block Properties

Many block properties use the *Path Selector* list to allow you to select a path or individual field as the value for the property. For example filter conditions let you select the comparison value for the condition from a field. You can select an input field to assign it to any block property that accepts paths or fields using the Path/Selector:

1. Add an Input block to the canvas.

2. Select the block where you want to use this input field for a property.
3. Click in the block property that accepts paths or use to open the Path Selector list and select the input field.

   Wires draws the connection between the Input block and the block you are currently working with. For example:



Format Results as CSV

The **CSVGenerator** action transforms document-type results into a comma-separated values (CSV) format for use in spreadsheet or database software. You must connect a document-type result as input to this action.

Note: The result of this block is a string, rather than a well-formed document.

Delimiter	Optionally, change the default delimiter to use to separate data for each column.

Decorate or Transform Data

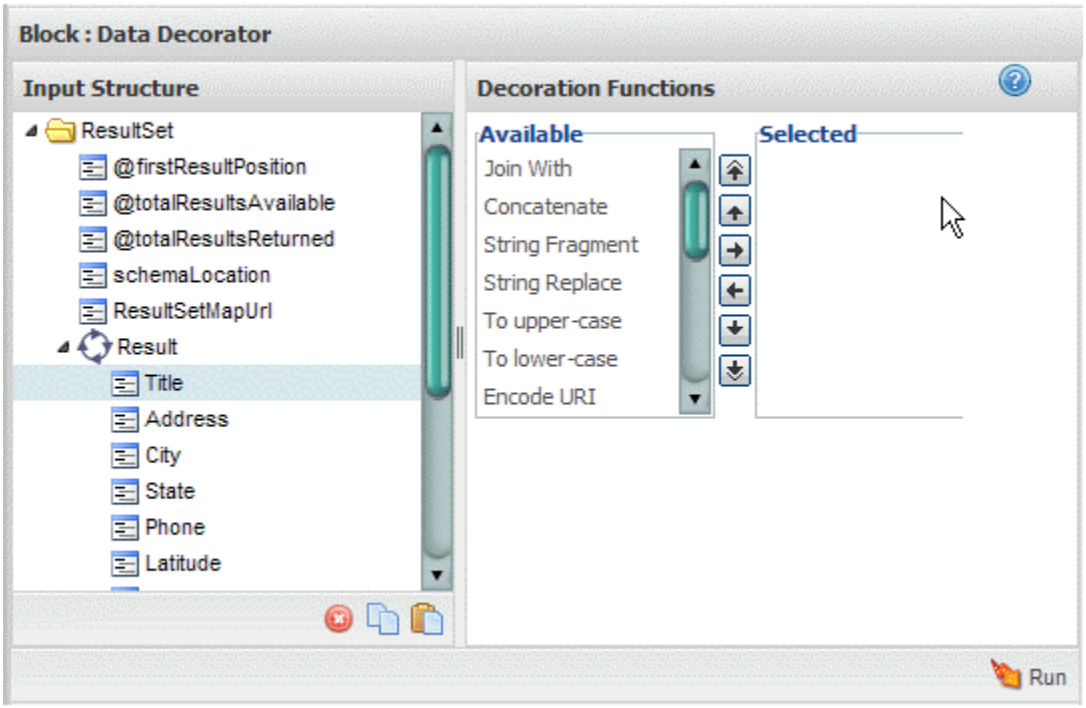
The **DataDecorator** action allows you to convert, calculate or transform just the data in results using one or more functions. You add functions to individual fields, as needed. With multiple functions, each is applied in order to the data for that field.

Note: This action does not change the structure of results, filter results or combine data in any way.

To transform result data

1. Connect document-type results from another block as input to **DataDecorator**.

   The Block Properties for this action displays two panes with the structure of the input document and a pane with properties you can assign to the currently selected input field.



   The input structure can contain:

   - = a root or structure node that contains children but does not repeat. The root node completely wraps the entire input document.
   - = a repeating node, usually with children.
   - = a field (node) that contains data. If the node name begins with `@` this is an XML attribute. Otherwise it is an XML element.

2. Select a field with data you want to convert.
3. Select the first function to apply to this data and click . Complete any properties needed for this function.

   MashZone NextGen has a set of built-in functions. Your MashZone NextGen administrator can also add functions to this list. For information on completing any of the built-in functions, see:

"Absolute" on page 360 | "Multiply" on page 364 | "To Datetime" on page 368

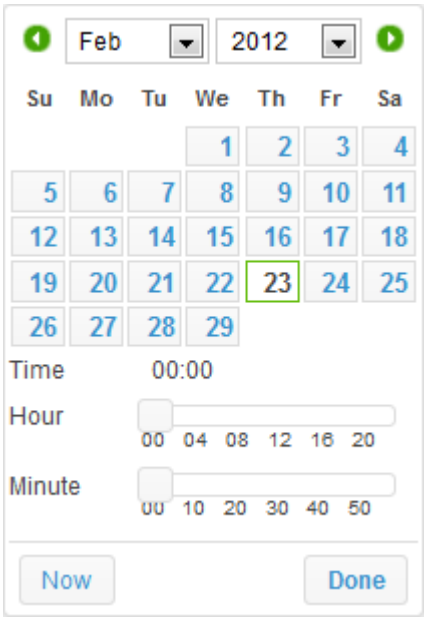
"Add" on page 360	"Round" on page 365	"To Decimal" on page 368
"Add to Date" on page 361	"Set Decimal Places" on page 365	"To Integer" on page 368
"Ceiling" on page 361	"String Replace" on page 365	"To Lower-case" on page 368
"Currency Value" on page 361	"Substring After" on page 366	"To Number" on page 369
"Date Formatter" on page 362	"Substring Before" on page 366	"To String" on page 369
"Divide" on page 363	"Substring" on page 366	"To Upper-case" on page 369
"Encode URI" on page 364	"Subtract" on page 367	
"Floor" on page 364	"Subtract from Date" on page 367	
"Join With" on page 364	"To Date" on page 368	

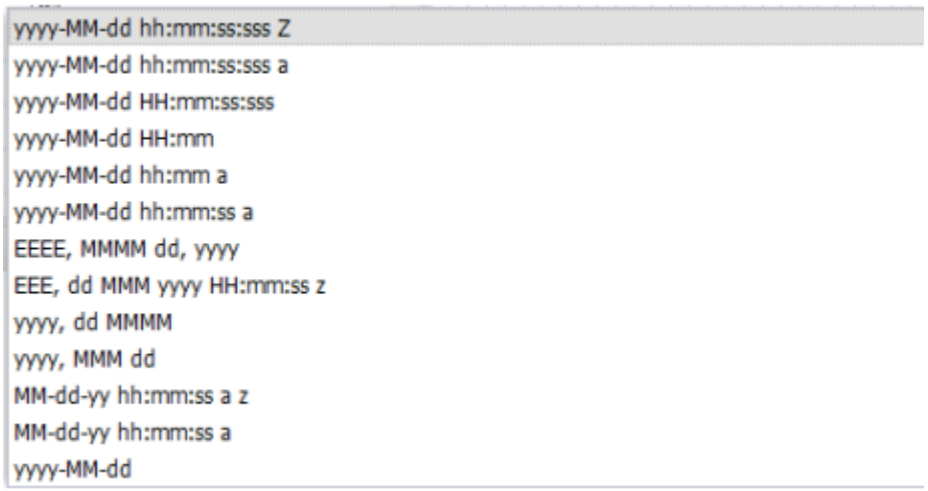
4. Complete the conversion for this field by:
   - Adding more functions, as needed, and completing their properties.
   - Changing the order in which functions are applied using and .
   - Using to clear any functions you have applied to a field.
5. Add conversions to other fields, as needed. Use **Preview** to see the effect of functions you have applied at any time.

Build Dates

The **DateBuilder** action allows you to select a date and optional time. You can also control whether this date is treated as a date or as a string.

Note: The result of this block is either a date, a date and time or a string rather than a well-formed document.

Input Date	Click in this field to select a date from the calendar that opens and optionally set a time. Click Now to pick the current date and time. 
Output Date Type	Select the data type to apply to this date: ■ `Date` = the value for this date and time, if any, are treated as a date. It can be used in date calculations or sorting by date. ■ `String` = the value for this date and time, if any, are treated as a string of text. You cannot use this value in date calculations or sorting. You can choose the format to use to display this date string.
Output Date Format	Select the format to apply to this date string. Date formats include:



Date and time syntax symbols includes:

- `a` = AM or PM for 12-hour times
- `dd` = 2-digit date of the day
- `EEE` = 3-character abbreviation of the day of the week, such as Thu
- `EEEE` = full name of the day of the week, such as Thursday
- `HH` = 2 digit hour in a 24 hour clock
- `hh` = 2 digit hour in a 12 hour clock
- `MM` = 2 digit month
- `MMM` = 3-character month abbreviation, such as Dec
- `MMMM` = full month by name, such as December
- `mm` = minutes
- `ss` = seconds
- `sss` = milliseconds
- `yyyy` = 4-digit year
- `z` = timezone

Create a Document for Input

The **Document** block creates a well-formed document that you can use to supply data for properties in other blocks.

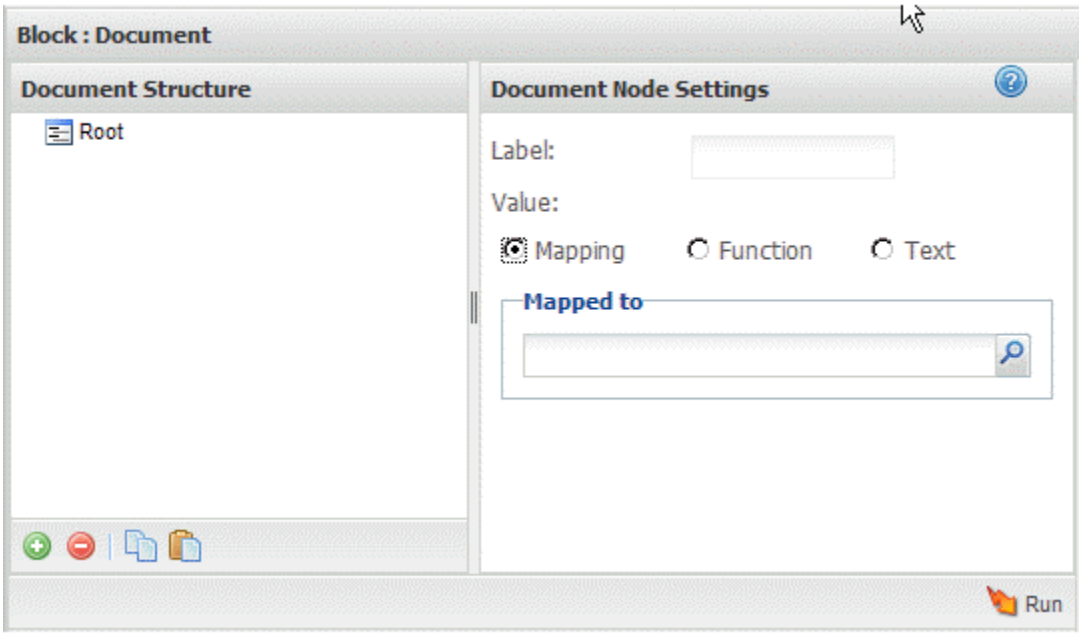
For example, some web services or web sites that you invoke using DirectInvoke require a username and password passed as a header. But DirectInvoke has only a single

property to define the header. You can use **Document** to build the complex document for this header and then assign it to the header property for the DirectInvoke block.

To define a document

1. Drag **Document** onto the canvas.

   The Block Properties for this action displays two panes with a pane where you build the document structure and a pane with properties you can assign to the currently selected node or field.



Tip: It helps to maximize the Block Properties/Preview panel with the button while you work with both structures.

   The output structure can contain:

   - = a root or structure node that contains children but does not repeat. The root node completely wraps the entire document of the results.
   - = a repeating node, usually with children.
   - = a field (node) that contains data. If the node name begins with `@` this is an XML attribute. Otherwise it is an XML element.

2. Change the name for the root node, if desired, in the Document Node Settings field.
3. Build the output document under the root node and define what data should populate the results: You can:

   - "Add Fields or Structure Nodes" on page 313 for more control over the structure and order. Then use node properties to populate data.

- "Map Other Results to the Output" on page 313 to populate data in the output structure.
- "Enter or Calculate Data" on page 314 for output fields. You can enter literal data or use functions to calculate data.
- *Delete output nodes*: select the output node and click .
- *Rename nodes*: enter a new name in the Document Node Settings field.

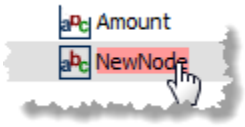
  Node names must be valid XML names, starting with a letter and containing only letters, numbers, periods(.), underscores (_) or dashes (-).
- *Copy/paste node property settings* from one output node to another with the and buttons in the Output Structure toolbar.

4. Use **Run** to preview the results as needed.

Add Fields or Structure Nodes

Use the toolbar to add nodes one at a time to the output document and then map them, assign values or calculate values as needed to define the data for this document.

1. Select the output node that should contain the new node.
2. Click to add a new, unmapped node.



3. To change the default name for the new node, double-click the node and enter a new name.

   To make the new node an attribute of its parent, start the name with `@`, such as `@id`.

   To make this a repeating node, keep adding new nodes with the same name.
4. Either:
   - Add new nodes under this to make this a structure node.
   - "Map Other Results to the Output" on page 313 or "Enter or Calculate Data" on page 314 for the node if it should be a data field.

Map Other Results to the Output

Mapping allows you to take data from other blocks and populate the result document.

1. Select the output data field you want to map.
2. Set the **Mapping** option in the Output Node Settings panel.
3. Click to open the **Path Selector** list and select:
   - A data field from the results of any other block in the mashup.

- An existing input field for the mashup.
- Add a new input field.

You can map single fields to single fields in the output. Similarly, you can map repeating fields to single fields in the output. You can also map entire documents to output fields. The result will be:

From > To	Result
	Single input value populates the single output field.
	The values from all repeating input fields are joined, in order, with spaces between the values and this single joined value populates the output field.
Document >	The full structure of the input document is appended as a child of the single output field.

Enter or Calculate Data

In addition to mapping, you can enter literal values for output fields or use functions to calculate or transform data.

1. Select the output data field you want to map.
2. To provide a literal value, set the **Text** option in the Output Node Settings panel and enter the value in the Text field.
3. To use a function to calculate a value, set the **Function** option in the Output Node Settings panel and:
   - Select the function from the list. MashZone NextGen has a set of built-in functions, but MashZone NextGen developers or administrators can add functions to this list.
   - Complete the properties for the function. See "Built-in Functions for Decorator, Mapper and Transformer Blocks" on page 359 for property information for the built-in functions.

Adapt Results as a SharePoint External Content Type

The **External Content Type Adapter** block enables the output of a mashup to work with a SharePoint*external content type* so that mashup results can be dynamically included in SharePoint sites as *external content lists*. This integrates mashup results in SharePoint sites supporting deep integration in many SharePoint features and services.

Mashups that use the **External Content Type Adapter** pull in results from other mashables or mashups, filter or transform these results if needed and then use the adapter block to enable external content type connections. This integration requires:

- Support for the *two*SharePoint operations required for external content types. See "Supported SharePoint Operations" on page 315 for details.
- Results that meet the "Data Requirements for the Results of ECT-Enabled Mashups" on page 316.
- And specific limitations on the "Updates to the Mashup's Results" on page 317.

Note: The **External Content Type Adapter** block is not necessarily required to allow SharePoint to connect to mashups as external content types, but it does simplify the integration in most cases.

Create an ECT-Enabled Mashup

To create an ECT-enabled mashup with this block

1. Add a mashable or other mashup as the source of external information for this mashup. Or use **DirectInvoke**, **SQL** or a custom block to supply this information.
2. If needed, use other actions to filter or transform these results. This may be needed to meet your requirements for the data you want in SharePoint or to meet "Data Requirements for the Results of ECT-Enabled Mashups" on page 316 such as limiting the number of items for the list based on this mashup.
3. Add an **Input** block and *change the name* for this input to match the name of the field in the information source that is the ID for each row.

   Important: The name and datatype of this input *must* match the name and datatype of the ID field exactly. See "Data Requirements for the Results of ECT-Enabled Mashups" on page 316 for more information.

4. Add the *External Content Type Adapter* block and:
   a. Complete the "ECT Block Properties" on page 317.
   b. Connect this block to the **Output** block.
   c. Save this mashup and turn it on.

Supported SharePoint Operations

The **External Content Type Adapter** allows mashups to support the following SharePoint list operations required for external content lists:

- *Read List*: to return all items for the list. This case simply returns the entire results of the mashup.
- *Read List Item*: to return a single list time by ID. This case requires an input parameter to specify the ID of the specific row in mashup results to return.

Mashups have only one operation to run the mashup and return all results. To support these two separate operations for SharePoint lists from one mashup, the *External Content Type Adapter* block requires an mashup input parameter for an item ID.

- An item ID of `0` matches the case where all list items must be returned to SharePoint.
- Any other item ID matches the case where a specific list item should be returned.

This in turn sets some "Data Requirements for the Results of ECT-Enabled Mashups" on page 316.

In most cases, the other operations available for external content lists in SharePoint to create, update or delete list items *cannot* be supported by ECT-enabled mashups because the underlying information sources (mashables or mashups) used by ECT-enabled mashups do not support these operations.

Data Requirements for the Results of ECT-Enabled Mashups

To enable integration of mashup results as SharePoint lists, the results of the mashup must:

- Have a set of repeating items (rows) that transform to the SharePoint list items.
- Each item (row) in the result must have a flat set of fields that each contain simple data. Complex content such as this example is not supported:

```
<row>
  <name>A field with simple content</name>
  <location>
    <street>complex field</street>
    <city>with </city>
    <state>individual</state>
    <country>children</country>
  </location>
  <phone>another field with simple content</phone>
</row>
```

- The data type for the content of each field must match a supported SharePoint list field type. See SharePoint documentation for specific information.

  Note: Valid field types includes the built-in SharePoint field types and optionally custom types added to your SharePoint environment. For information on types that are *not* supported, see Datatypes Not Supported by Business Data Connectivity Service.

- Each row must have a required field that can be used as a unique identifier (the ID) for that row. This identifier cannot change for the lifetime of that item.

  Numeric row IDs cannot exceed 12 digits.

- Row IDs that are simple indexed numbers (integers) are *strongly* recommended as this field must match the required input parameter for the *External Content Type Adapter* block. This is used to support the two required operations for SharePoint external content lists. See "Supported SharePoint Operations" on page 315 for more information.

- Not exceed the maximum number of rows allowed. This maximum is set by your SharePoint Farm Administrator based on resources specific to your environment.

In addition, the name and datatype of the input parameter that is required for the *External Content Type Adapter* block must match the name and datatype of the field used as the row ID in mashup results.

ECT Block Properties

Select the ID field	Click to open the Path Selector list and select the field in the repeating-items from the input to this block that is the required unique identifier for each item (row). See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. This field becomes the item ID for the items in external content lists in SharePoint based on this mashup. See "Data Requirements for the Results of ECT-Enabled Mashups" on page 316 for information on ID requirements for this field.
Select the list item ID input	Click to open the Path Selector list and either: ■ Select the → **Input** block for item IDs that you already created for this mashup. ■ Click **Add as an Input** to create an input parameter for this mashup. Then *change the name* for this input parameter to match the name of the field in the information source that is the ID for each row. The name and datatype of this input *must* match the name of the ID field exactly. See "Data Requirements for the Results of ECT-Enabled Mashups" on page 316 for more information. SharePoint uses this parameter to either identify the specific list item to return (Read List Item operation) or to specify `0` (Read List operation) to return all list items for external content lists based on this mashup.

Updates to the Mashup's Results

SharePoint external content types define both how to run an ECT-enabled mashup and information about the parameters for and results from that mashup. This is defined in a WSDL (web-service definition) when the external content type is created.

Because of this, changes to the parameters for an ECT-enabled mashup or the number, name or types of data for each field in its results affect the information in the mashup's

WSDL. Any of these change can cause the external content type to break and cause errors in the related external content lists in SharePoint.

To fix problems caused by changes to the results or parameters for an ECT-enabled mashup, you must delete and recreate the external content type and related external content lists for that mashup in SharePoint Designer.

Extract Partial Results

The *Extract* action extracts partial results, either as a single value or as a well-formed document. You can use partial results as input to another block or to a specific block property. You can:

- *Extract a Value*: from one, non-repeating field. The extracted result is that single value.
- *Extract the Values for One Field in a Set of Repeating Items*: the extracted result is a single value that combines the values of all occurrences for that field, in order, separated by commas.
- *Extract a Structure or a Set of Repeating Items*: for a single structure, the partial result is that node and all its descendants, as a well-formed document.

  The partial result for a set of repeating items is those selected items wrapped in a new root node, creating a well-formed document. You can define the root node to use and select specific occurrences.

Pick a path to be extracted	Click in this field to pick the specific field, structure or set of repeating items you want to extract: ■ If you select a field with data, the partial result is a single value. If the field you select is in a set of repeating items, the resulting value combines the values of all occurrences of that field, separated by commas. ■ If you select a non-repeating structure, the result is that structure. ■ If you select a repeating item or a structure within a set of repeating items, use the additional properties to define exactly what to extract. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Result Type	This optional property is only available if you have selected a field with data to extract, either as a single field or a field in a set of repeating items. Use this property to indicate the type of data in this field, if the default type is incorrect. You can choose from: ■ `boolean` for true/false data. ■ `datetime` for dates, times or combinations in one of the valid date or time formats. See "Date and Time Formats, Math and Sorting in Wires" on page 386 for more information. ■ `date` is *deprecated*. Use `datetime` instead. ■ `decimal` for decimal numeric data for mashables. ■ `integer` for integer numeric data for mashables. ■ `number` for numeric data of any type for mashups. ■ `string` for text data.
Result Name	This optional property is only available if you have selected repeating items or a structure within repeating items to extract. Use it to define the name of the root node to wrap the extracted results. The value must be a valid XML name, beginning with a letter and containing letters, numbers, period, underscore (_) or dashes (-). No spaces or other punctuation are allowed. If you leave this property blank, the root node has a default name of `result`.
Extract	Use these options to define exactly which repeating items to extract. The default is `All`, but you can select a single item or a range of items.

Filter Results

The **Filter** action selects specific repeating items from a result based on matching characters or on meeting a numeric relationship. Connect the block with document-type results that you want to filter as input to this action.

Note: If there are no matching items, **Filter** return an empty document that typically contains just the root node with no repeating items or data.

apply condition	If you set multiple filter conditions, use this option to determine whether all conditions or only one must be met for an item to be selected in the filtered results.
left filter condition field	Click the Path Selector button to find the field in each repeating item that you want to use to filter results. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. Or click in this field and begin typing a field name. Select one of the suggested fields (with its full path).
comparison function	If needed, change the comparison function for the filter condition: ■ `Contains` = checks to see that the text characters you are filtering for are present somewhere in the field you have chosen to filter by. The comparison is not case sensitive, unless you set the Case Sensitive option. ■ `Does not contain` = checks to see that the text characters you are filtering for are *not* present somewhere in the field you have chosen to filter by. The comparison is not case sensitive, unless you set the Case Sensitive option. ■ `Is Empty` = checks to see that the field is present, but contains no value. This is also sometimes called *null*. ■ `Matches` = checks to see that the characters you are filtering for are present somewhere in the field you have chosen to filter. ■ `Does not match` = checks to see that the characters you are filtering for are *not* present somewhere in the field you have chosen to filter. ■ `Is Present` = checks to see that this field exists within the current row. ■ `Is Not Present` = checks to see that this field does not existing within the current row.

	■ `= (equals), != (does not equal), > (is greater than), < ()is less than, >= (is greater than or equals) or <= ()is less than or equals` = compares the entire value of the field you select with the value you set for the filter condition using standard numeric comparisons.
right filter condition field	Set the value to check for in the right side of the filter condition. You can: ■ Enter a value, word or phrase directly in the right side of the filter condition. **Note:** Filter conditions do not check for separate words unless you use wildcards. So a match for `Dow Jones` would look for both words together, but would not match a result containing the word "window." See "Filtering Techniques" on page 322 for tips on using wildcards. ■ Use an input field to provide the value dynamically. See "Selecting Existing Input Blocks for Block Properties" on page 306 or "Add Input Fields Directly to Block Properties" on page 303 for instructions. ■ Use a MashZone NextGen global or user attribute to provide the value. See "Use MashZone NextGen Global or User Attributes in Block Properties" on page 374 for instructions. **Note:** This property is not available when the type of comparison that you have selected does not require a right field.
Case Sensitive	Set this option if the condition should be case sensitive.
Add Condition	Add another filter condition.

Note: MashZone NextGen developers who know XPath well can use the *Advanced Mode* button to customize the XPath expressions for conditions. Use the advance mode with care, however, as it disables the display of conditions in basic mode. If you return the condition to basic mode, your customized XPath expression for the condition is lost.

See "Edit Mashup XPath Expressions in Advanced Mode" on page 391 for more information.

Filtering Techniques

The **Filter** action has two different ways to compare the contents of a field: contains and matches.

The **contains** comparison matches the contents of a field exactly to the characters you enter. You cannot use wildcards.

The **matches** comparison also matches the contents of a field to the characters you enter as a *match expression*. Results are considered a match as long as the expression occurs somewhere within the content of the field. However, the comparison is not case sensitive. And you can use wildcards to make the comparison more general or more specific.

Note: Match expressions are *regular expressions* - a technical and very flexible syntax for matching groups of characters. There are many additional features in regular expressions that you can use in comparisons with a matches filter condition which are not covered in this topic.

Some of the most useful wildcards that you can use in a match condition are shown below.

For	Use Wildards	Examples
Any single character	.	`bas.` would match "bas", "bass" or "base" but not "bases".
Any number of characters	.*	`jav.*` would match a field containing "Mojave", "Java", "javascript" or "Javier".
Field starts with	^	`^Dow` would match a field that began with "Dow Jones", but not one that contained "window". `Dow`, however, would match both a field that began with "Dow Jones" and one that contained "window".

If you need to match any of the wildcard characters themselves, use `\` before. So `Mr\.` would match "Mr." but would not match "Mrs".

Group Results

Use the **Group** block to group repeating items based on the unique values of one field. Optionally, you can filter items for a group or calculate simple statistics for the group. You can define multiple levels of groups.

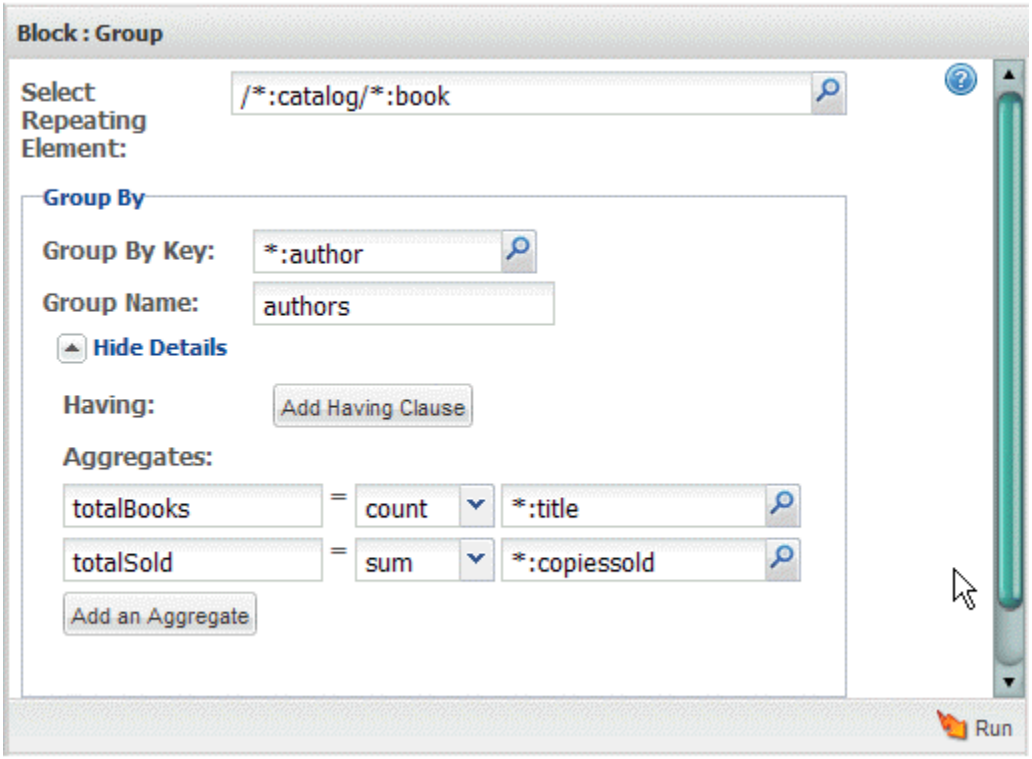
1. Connect a document-type result as input to the **Group** block.
2. Use the Path Selector in **Select Repeating Element** to identify the repeating items to be grouped.

   See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

3. In **Group**:
   a. Use the Path Selector in **Group Key** to find the field within these repeating items to use to group data for this level. Each unique value in this field will define one group.
   b. Optionally, enter a name for the node that should wrap all items for this group in **Group Name**.

      Note: Names must be valid XML names, starting with a letter and containing only letters, numbers, underscores (_), dashes (-) or periods (.).

      If you omit this property, the root node wrapping all groups defaults to `output`. For lower level groups, omitting this property omits the wrapping node.
   c. If needed, "Set Conditions to Filter Items to Include in Groups" on page 324' or "Add Calculations for the Group" on page 326.



4. If you need additional levels of grouping for these repeating items, click **Add Nested Group**.

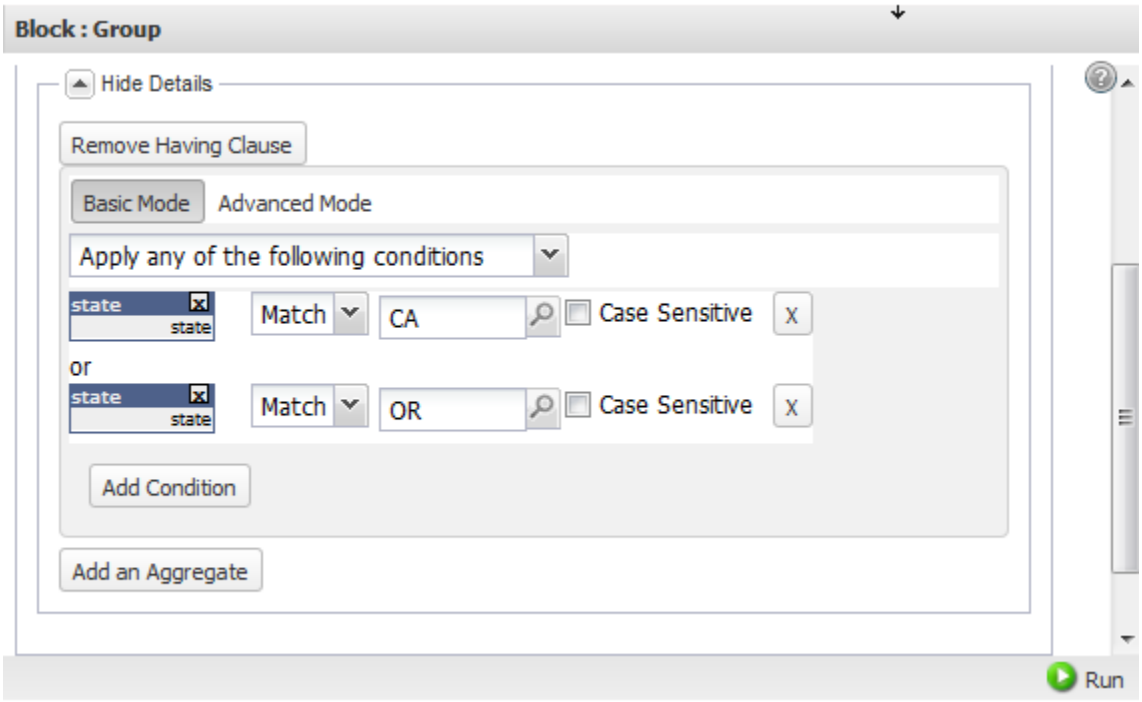
   This opens a new **Group** section. Complete this section as needed (see previous step) and continue adding **Group** sections to add additional groups as needed.

See "Group Configuration Example" on page 327 for an example of group configuration and the output.

Set Conditions to Filter Items to Include in Groups

To filter which repeating items are included in a group, you add a "having" clause.

1. Click **Show Details** in the **Group** section for that group, if details are not visible.
2. Click **Add Having Clause**. This adds fields defining the filter condition to use to include items in the group.



3. In the left field, use the Path Selector to find the field to compare for this filter. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
4. Choose the comparison operator.

   Use the basic math operators (=, <, <=, >, >=) for numeric fields. Use `Matches`, `Does Not Match`, `Contains` or `Does NotContain` operator for partial matches in text fields.
5. Enter a value to compare this field to or use the Path Selector to find the field for this comparison. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
6. If you need to filter items based on several conditions:
   a. Click **Add Condition** for another condition and complete the condition fields.
   b. Choose to **Apply any** or **Apply all** of the conditions.

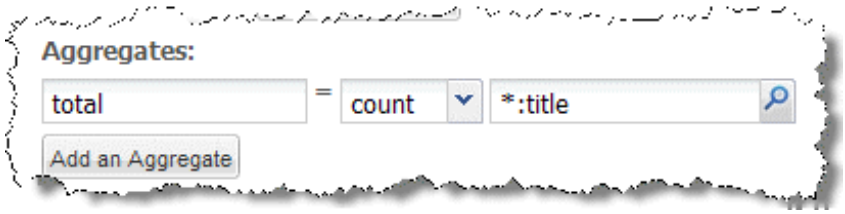
Note: MashZone NextGen developers who know XPath well can use the *Advanced Mode* button to customize the XPath expressions for conditions. Use the advance mode with care, however, as it disables the display of conditions in basic mode. If you return the condition to basic mode, your customized XPath expression for the condition is lost.

See "Edit Mashup XPath Expressions in Advanced Mode" on page 391 for more information.

Add Calculations for the Group

To add statistics or other calculations for items in a group

1. Click **Show Details** in the **Group** section for that group, if aggregates are not visible.
2. Click **Add an Aggregate**. This adds fields defining the equation for this calculation.



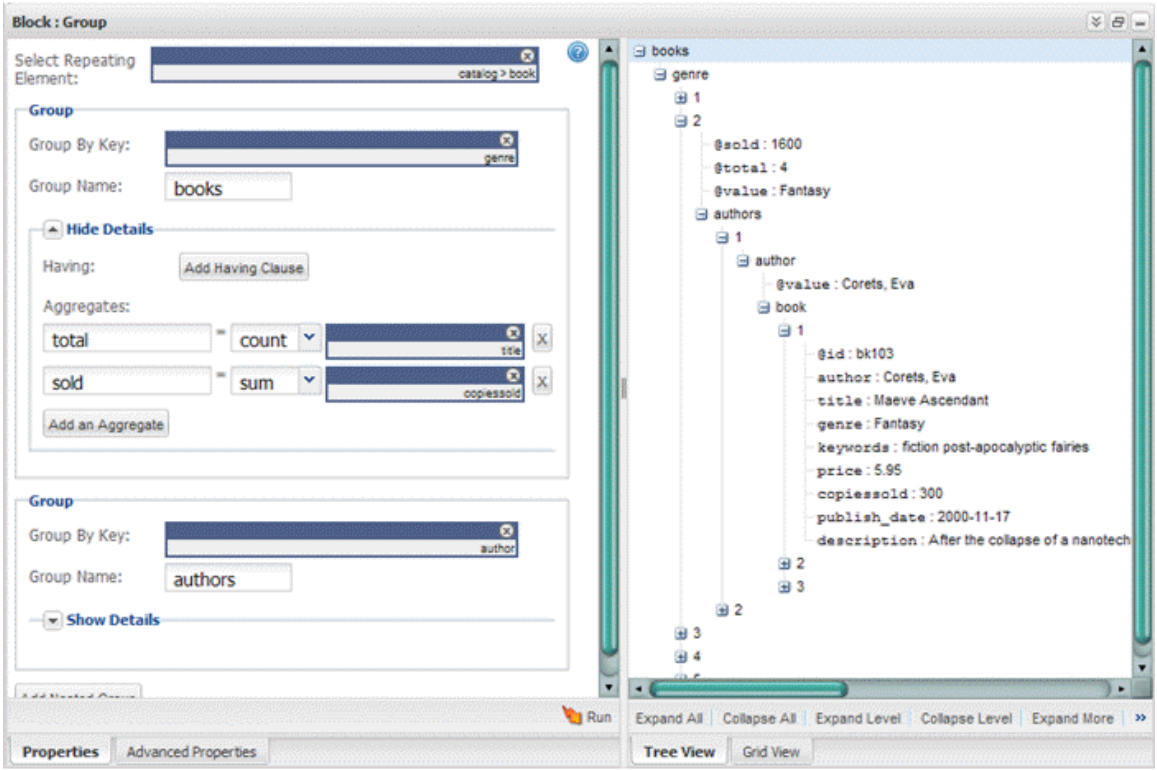
3. In the left field, enter a name for the result node to contain this calculation.

> **Note:** Names must be valid XML names, starting with a letter and containing only letters, numbers, underscores (_), dashes (-) or periods (.).

4. Choose the function for this calculation:
   - `count`
   - `sum`
   - `avg`
   - `min`
   - `max`
5. Use the Path Selector to find the field within these repeating items to use for this calculation. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

   Except for the `count` function, this field *must* have numeric data.
6. Add more calculations, if needed.

Group Configuration Example



Add Input Parameters

The ➔ **Input** action block defines an input parameter for the mashup. You and other users can use input parameters to dynamically control the information or behavior of the mashup. Mashup input parameters can assign values to virtually any block property within the mashup for any number of blocks.

There are several ways to add an ➔ **Input** action block and connect it a block property for another block:

- Automatically Add Input Fields for Block Properties
- Select an Existing Input Block for Block Properties
- Manually Add and Connect Input Blocks to Block Properties

See also "Changing Mashup Input Parameter Names" on page 374.

Name	Wires provides a default input parameter name when you add ➔ **Input** to a mashup. If this mashup has not yet been turned on, you may change the name of an input parameter.

	Important: *Never* change the name of a mashup input parameter once you have turned a mashup on. This can cause apps or other mashups that use this mashup to fail. Input parameter names must be unique within a mashup and valid. See "Valid Input Parameter Names" on page 375 for specific details.
Data Type	Optionally, choose the type of data that goes in this input field: ■ `boolean` for true/false data. ■ `datetime` for dates, times or combinations in one of the valid date or time formats. See "Date and Time Formats, Math and Sorting in Wires" on page 386 for more information. ■ `date` is *deprecated*. Use `datetime` instead. ■ `decimal` for decimal numeric data for mashables. ■ `document` for complex input with a hierarchy. The data must be a *well-formed XML document*. ■ `integer` for integer numeric data for mashables. ■ `number` for numeric data of any type for mashups. ■ `string` for text data.

Default Value	Optionally, enter a default value for this input field.
	
	For document-type parameters, you can enter a default value as a *well-formed* XML document. The entire structure must be contained within one XML element. For example:
	`<name>`
	`  <first>Ethan</first>`
	`  <last>Fromme</last>`
	`</name>`

Automatically Add Input Fields for Block Properties

You can automatically create a new input parameter for the mashup, using the *Path Selector* list, and connect it to most block properties, including input parameters for information sources or action block properties.

1. Click in a block property field or click to open the **Path Selector** list.
2. Click **Add as input**.

   This adds an **Input** block and automatically connects it to the block property.
3. If needed: Or change the input parameter name and label. See "Changing Mashup Input Parameter Names" on page 374.

- Set the datatype or default value for the input parameter.
- Or change the input parameter name and label. See "Changing Mashup Input Parameter Names" on page 374 for more information.

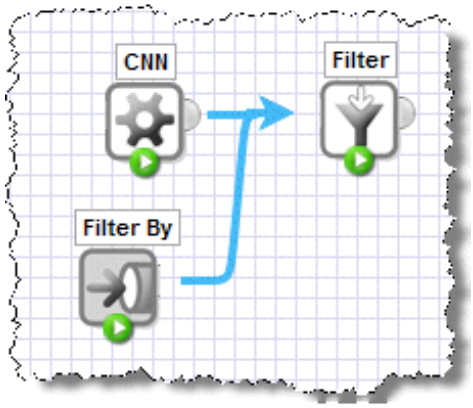
Select an Existing Input Block for Block Properties

You can also select an existing input parameter for the mashup and automatically connect it to most block properties using the *Path Selector* list.

For example, filter conditions let you select the comparison value for the condition from an input field. You can select an input field to assign it to any block property that accepts paths or fields using the Path/Selector:

1. Click in a block property field or click to open the **Path Selector** list.
2. Select the existing input parameter from the list.

   Wires draws the connection between the Input block and the block you are currently working with. For example:

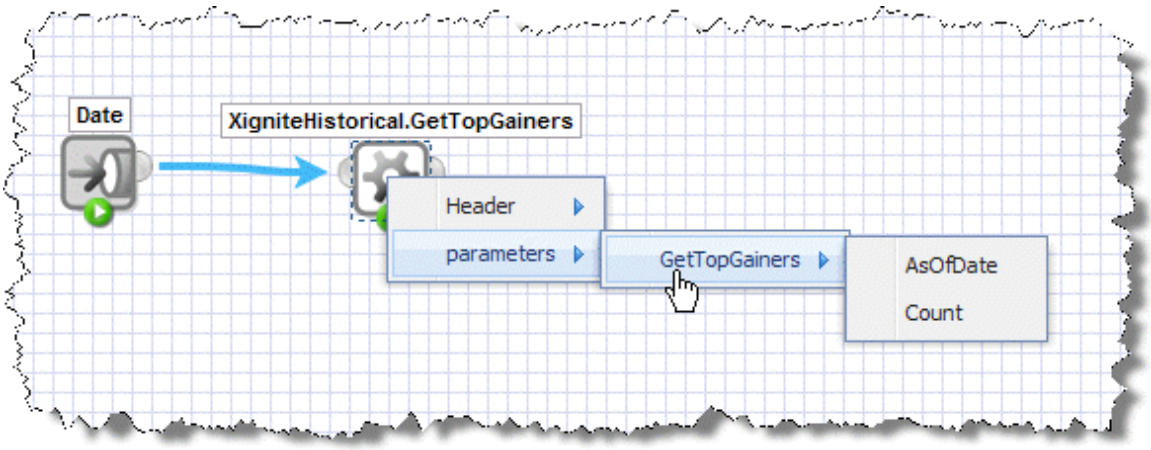


3. If needed, set the datatype or default value for the input parameter. Or change the input parameter name and label. See "Changing Mashup Input Parameter Names" on page 374.

Manually Add and Connect Input Blocks to Block Properties

Like any action block you can manually add this block to the Wires canvas and connect it to another block, selecting the block property it should be connected to.

1. Add an **Input** block to the canvas.
2. If needed, set the datatype or default value for the input parameter. Or change the input parameter name and label. See "Changing Mashup Input Parameter Names" on page 374.
3. Draw a connection from **Input** to the block with the property you want to have set by this input.
4. Choose the block property to assign this input parameter to.



As this example shows, properties may also be in a 'header' for the mashable information source or mashup.

Join Results

The **Join** action combines repeating sets of *records* from two results based on a relationship that you define between fields in each record.

You define how records are joined with one or more *join conditions*. You can also define how multiple conditions are used: either all conditions must be matched or at least one must be matched.

This acts somewhat like a join for a database. The results include only those items from the two inputs that match one or more of the join conditions. If a matching item from the first input does not have a matching item from the second input, it is not included, and vice versa.

You must connect two blocks as the input to **Join** and define at least one join condition. You can also use the following advanced techniques for **Join**:

- With **Join** blocks created in version 3.5 or later, you can also "Fine Tune the Joined Result" on page 331.

  As of version 3.5, this action block uses a different method to perform joins than in earlier releases. This change has no effect on mashups from earlier releases.

  If you wish to find tune fields and values for the joined result in mashups from earlier releases, see "Upgrading Mashups with Join Blocks from Previous Versions" on page 334 for instructions.

- MashZone NextGen developers who know XPath well can use the *Advanced Mode* button to customize the XPath expressions for conditions. Use the advance mode with care, however, as it disables the display of conditions in basic mode. If you return the condition to basic mode, your customized XPath expression for the condition is lost.

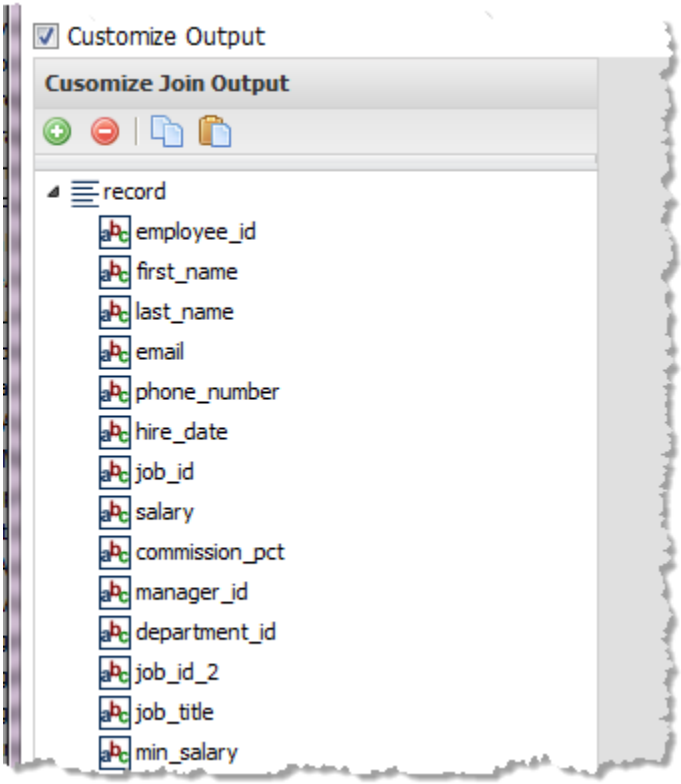
See "Edit Mashup XPath Expressions in Advanced Mode" on page 391 for more information.

apply option	If there are two or more join conditions, set this option to determine whether only one condition is sufficient to ensure a join or all conditions must be met.
join condition	Each join condition has a left- and right-field and a relationship operator: ■ *left-field*: click in this field to select the field from one result that should be used to define this join condition. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. ■ *operator*: select the relationship operator for this join condition. ■ *right-field*: click in this field to select the field from the remaining result that should be used to define this join condition. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Case Sensitive	Set this option if this join condition should be case sensitive.
Add Condition	Use this button to add more join conditions.
Customize Output	Set this options if you need to add or delete fields from the joined record or apply functions to the data in fields. See "Fine Tune the Joined Result" on page 331 for instructions.

Fine Tune the Joined Result

For **Join** blocks created in version 3.5 or later, you can edit the structure of the joined result, adding or deleting fields as needed, updating mapping information or use functions to fine tune the end result:

1. Set the **Customize Output** option. This opens a new section in the block properties that shows the generated structure of the joined items.



The joined structure can contain:

- = a root or structure node that contains children but does not repeat. The root node completely wraps each joined item of the results.
- = a repeating node, usually with children.
- = a field (node) that contains data. If the node name begins with `@` this is an XML attribute. Otherwise it is an XML element.

2. Update the structure, names and mapping as needed. You can:
   - Add Fields or Structure Nodes for more control over the structure and order. Then use node properties to populate data.
   - Enter or Calculate Data to populate data in the output structure.
   - Map Other Results to the Output for output fields. You can enter literal data or use functions to calculate data.
   - *Delete output nodes*: select the output node and click .
   - *Rename nodes*: double-click the field and enter a new name.

     Node names must be valid XML names, starting with a letter and containing only letters, numbers, periods(.), underscores (_) or dashes (-).

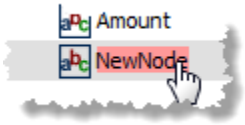
     To make the node an attribute, start the name with `@`.

- *Copy/paste node property settings* from one output node to another with the and buttons in the Join Output toolbar.

Add Fields or Structure Nodes

Use the toolbar to add nodes one at a time to the joined output and then map them, assign values or calculate values as needed to define the data for this joined result.

1. Select the output node that should contain the new node.
2. Click to add a new, unmapped node.



3. To change the default name for the new node, double-click the node and enter a new name.

   To make this new node an attribute of its parent, start the node name with `@`, such as `@id`.

   To make this a repeating node, keep adding new nodes with the same name.
4. Either:
   - Add new nodes under this to make this a structure node.
   - Map Other Results to the Output or Enter or Calculate Data for the node if it should be a data field.

Map Other Results to the Output

Mapping allows you to take data from other blocks and populate the joined result.

1. Select the join output field you want to map.
2. Set the **Mapping** option.
3. Click to open the **Path Selector** list and select:
   - A data field from the results of any other block in the mashup.
   - An existing input field for the mashup.
   - Add a new input field.

You can map single fields to single fields in the output. Similarly, you can map repeating fields to single fields in the output. You can also map entire documents to joined output fields. The result will be:

From > To	Result
>	Single input value populates the single output field.
>	The values from all repeating input fields are joined, in order, with spaces between the values and this single joined value populates the output field.
Document >	The full structure of the input document is appended as a child of the single output field.

Enter or Calculate Data

In addition to mapping, you can enter literal values for output fields or use functions to calculate or transform data.

1. Select the joined output field you want to map.
2. To provide a literal value, set the **Text** option and enter the value in the Text field.
3. To use a function to calculate a value, set the **Function** option and:
   - Select the function from the list. MashZone NextGen has a set of built-in functions, but MashZone NextGen developers or administrators can add functions to this list.
   - Complete the properties for the function. See "Built-in Functions for Decorator, Mapper and Transformer Blocks" on page 359 for property information for the built-in functions.

Upgrading Mashups with Join Blocks from Previous Versions

In version 3.5, the way **Join** block joins results was updated to better handle cases where the input has a complex structure or includes attributes. Mashups created in versions prior to 3.5 will continue to join results in the same manner they always have, so this update does not disrupt your existing mashups.

This updated method also supports fine tuning the structure and mapping for the joined results. To use the fine tuning feature for **Join** blocks in existing mashups, you must delete the existing **Join** block(s) and recreate them.

Map Results to Known Structures

The **Mapper** action allows you to change the structure and data from document results to a specific, well-known document structure that you chose. MashZone NextGen administrators must add document structures to this list.

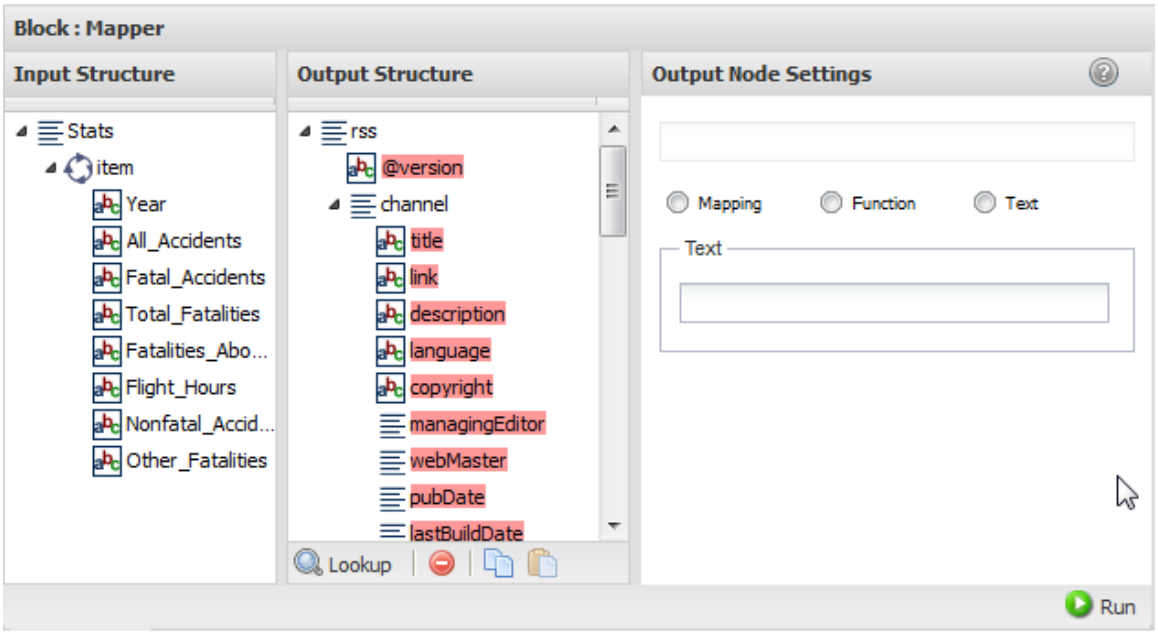
To define the transformation

1. Connect document-type results from another block as input to **Mapper**.
2. Click **Choose output structure** and choose:
   - Choose **From active services > *mashable/mashup name*** if the structure you need to map to is used by a mashup or mashable block in this mashup.
   - Choose **From schema files > Load schemas** if the structure you need to map to is defined by an XML schema.

     Note: MashZone NextGen administrators must add relevant schemas for this option to be available.

     Then choose **From schema files > *schema-name*** for the schema you need and select the root node for this new document.

   The Block Properties for this action displays three panes with the structure of the input document, a pane where you choose the well-known output structure that you want to map to, and a pane with properties you can assign to the currently selected output node or field.



Tip: It helps to maximize the Block Properties/Preview panel with the button while you work with both structures.

Both the input and output structure can contain:

- = a root or structure node that contains children but does not repeat. The root node completely wraps the entire document of the results.
- = a repeating node, usually with children.
- = a field (node) that contains data. If the node name begins with `@` this is an XML attribute. Otherwise it is an XML element.

3. For each output data field, either:
   - Map a data field from the input document to a data field in the output document.
   - Map an input field from the mashup, add an input field to the mashup and map that, or map any data field from the results of other blocks.

     Select the output data field you want to map. In the Output Node Settings pane, set the **Mapping** option. Click to open the **Path Selector** list and select the data field or mashup input field to map to this output.
   - Enter a literal value or calculate a value using a function. See "Enter or Calculate Data" on page 337 for instructions.

   With mapping, you can map single input fields to single output fields or to repeating output fields. Similarly, you can map repeating input fields to single output fields or to repeating output fields. You can also map entire documents to output fields. The result will be:

From > To	Result
>	Single input value populates the single output field.
>	Single input value populates all the repeating output fields.
>	The values from all repeating input fields are joined, in order, with spaces between the values and this single joined value populates the output field.
>	Each repeating item from the input document generates a corresponding repeating item in the output. Data fields from a repeating input item populate the mapped field in the repeating output item.
Document > or	The full structure of the input document is appended as a child of the single output field or all repeating output fields.

You can also copy and paste node property settings from one output node to another with the and buttons in the Output Structure toolbar.

4. Use **Run** to preview the results as needed.

Enter or Calculate Data

In addition to mapping, you can enter literal values for output fields or use functions to calculate or transform data.

1. Select the output data field you want to map.
2. To provide a literal value, set the **Text** option in the Output Node Settings panel and enter the value in the Text field.
3. To use a function to calculate a value, set the **Function** option in the Output Node Settings panel and:
   - Select the function from the list. MashZone NextGen has a set of built-in functions, but MashZone NextGen developers or administrators can add functions to this list.
   - Complete the properties for the function. See "Built-in Functions for Decorator, Mapper and Transformer Blocks" on page 359 for property information for the built-in functions.

Merge Results

The **Merge** action combines results from two or more blocks. The structure of the results being merged must be identical, with the exception of the root node for each result, or a selected, repeating portion of all results must be the same.

Note: If the blocks results you need to merge are SharePoint Lists, use the Merge SharePoint Lists block instead.

Before you can assign properties, connect two or more blocks as input to **Merge**. The properties you can set for this block depend on the format of the sources you are merging. See "Properties for Web Feed Sources" on page 337 or "Properties for Sources That Are Not Web Feeds" on page 338.

Properties for Web Feed Sources

Feed Type	Choose the standard format that is used by all web feed sources that you are merging. This defaults to `RSS`. When merging web feeds, all sources must be either RSS or Atom formats. They may be different versions of a particular standard format.
Title	Optionally, provide a feed title for the merged results. This title is frequently used in views for web feeds to identify the

	source and general types of new articles or blogs that appear within the feed.
Author	Optionally, provide a description of the combined authors for this merged web feed. This helps to identify all of the organizations that provided articles.
Description	Optionally, provide a general description for the merged web feed. This can appear in views for the feed and helps to clarify the types of articles that may appear within the merged results.
Link	Optionally, provide a link for the merged web feed. This can appear in views for the feed.

Properties for Sources That Are Not Web Feeds

Name for Root Node	Optionally, change the name to use for the root node of the merged results. If you do not set this, the root name for the first result being merged is used.
Merge Method	■ `Merge matching records for all inputs`: if the structures of the results being merged are not identical, but there is an identical repeating item (the *record*) in all of the results being merged, set this option to select that common repeating item. The repeating items are merged, but data in previous levels is not, thus potentially losing some data. ■ `Make each result appear as-is under results`: if the structures of the results being merged are not identical, this option appends each result, as-is, as siblings under a single root node. ■ `Merge on specified XPath (Advanced)`: if there is an identical node in all the results being merged, you can use this method to define what portion of the results should be appended from each result. This is useful if the nodes you want to merge are not repeating. **Note:** To use this method, you must be familiar with XPath syntax.
Records in source	If you are using the `Merge matching records for all inputs` method, this field shows the name of the repeating item selected as the records to merge. If the results have

	several different repeating items, you can change which items should be used as merged records.
Merge Path	If you are using the `Merge on specified XPath (Advanced)` method, enter the XPath expression, starting from the root node, that identifies the node or nodes in each result being merged to append in the merged results. For example: `/doc/students/student/address` You may select repeating or non-repeating nodes with this XPath. Developers should see "A Brief Introduction to XPath 2.0" on page 596 for more information.

Merge SharePoint Lists

The *Merge SharePoint Lists* block merge the results from two or more SharePoint Lists.

Note: If the blocks results you need to merge are *not*SharePoint Lists, use the Merge Results block instead.

Before you can complete properties for this block, connect the results of two or more *SharePoint List* blocks as input to this block.

Generate a column with unique ID	This property is set by default. It allows MashZone NextGen to add a field to each List item and assign a unique ID to each. This ensures that duplicate item IDs, if any, from the various SharePoint Lists you are merging are not a problem.
If merging more than one SharePoint task list, generate a column with unique predecessor?	This property is only required if there is more than one input block with results from a SharePoint Task List. For task lists, each task has a predecessor indicating task dependencies. Merging task lists can result in predecessors being erroneously merged as the predecessors are not unique. To fix this problem, you must set this option and then select the field that identifies task predecessor for each input block with task list results.
Select which field is the predecessor for each input block	These properties are only available if you have set the option to generate unique predecessors. One property is available for each input block you are merging. For those input blocks whose results are task lists, select the field in the results to identifies task predecessors (dependencies). MashZone NextGen will add a column with

	unique predecessor information for the merged list and use the predecessor information for each input block to properly assign dependencies.

Strip Namespaces

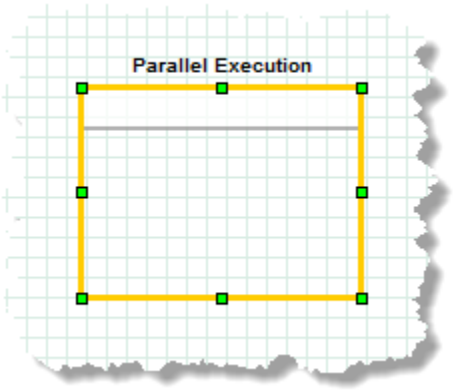
The **NamespaceStripper** action removes all namespaces from the results of one block. Removing namespaces can make it easier to work with results, depending your goals for the mashup. Simply connect a block with a document-type result that contains namespaces.

Run Several Blocks Simultaneously

Use the **Parallel** action block to run two or more mashups, mashables or action blocks at the same time in your mashup. Typically, this helps to improve performance for the mashup.

Note: Each block in **Parallel** runs independently of all other blocks within the loop. Thus, you can *only* wire blocks in **Parallel** to blocks that are outside of **Parallel**.

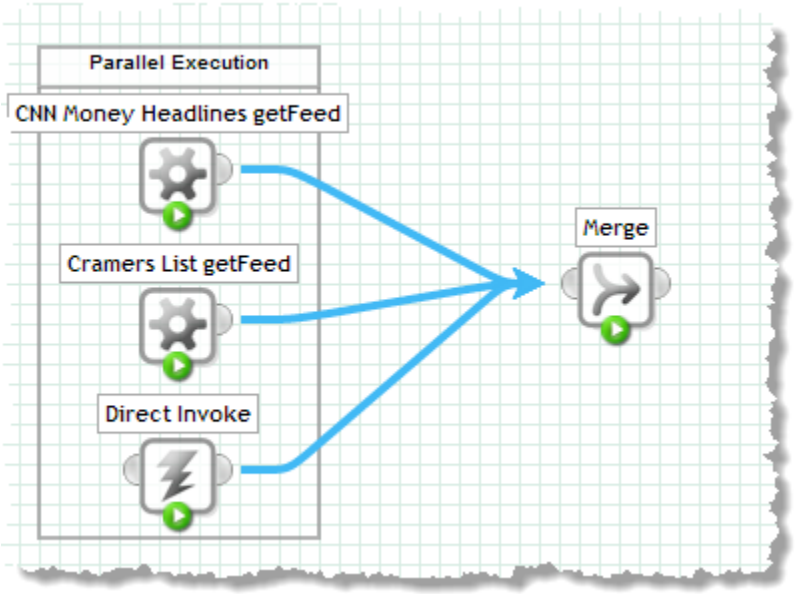
1. Add **Parallel** to the canvas. This defines a section of the canvas to use to hold the blocks you want to run simultaneously.



2. Add mashable, mashup or action blocks and drop them into the canvas area for **Parallel**.

   The area of the canvas that Parallel encompasses automatically extends as you add blocks.

3. Add the inputs or outputs to these blocks outside **Parallel**. Draw connections between blocks outside **Parallel** to blocks inside **Parallel** just as you normally would.



Pivot Data

The **Simple Pivot** action allows you to ?aggregate calcs based on distinct values of one field and group by another ? flattened to simple records

group results into two levels, get an aggregate calculation on subgroups (the second level) and flatten the result into a simple "table" of records and fields. This is similar to *pivot tables* in spreadsheets.

With results in the following form, for example:

```
records
 + record
   - salesperson
   - item
   - region
   - qty
```

You could use the **Group** action to get subtotals by region for each item. The results would have one record for each unique `item-region` combination. If you group on the `item` field and subgroup by `region` the results look something like this:

```
records
 + record
   - salesperson
   - item
   - north
   - east
   - south
   - west
   - central
```

The columns for `north` through `central` each contain the subtotal for that region.

Pick a record table to pivot data	The repeating record that identifies the table of data to pivot. This defaults to the first repeating record in the results you have connected as input to this block. Click in this field to select a different repeating record. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Pick a field to group records on	Click in this field to select the field for the top level grouping of this pivot table. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. The unique values for this field are used to group records from the results.
Pick a field to generate column headers	Click in this field to select the field for the second level of grouping for this pivot table. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. Each unique value of this field adds a column to the result records for this block with the aggregate calculation for that subgroup. The new column name is the corresponding unique value. **Note:** Actual column names may not match the unique values exactly because they must meet XML name standards. Wires automatically alters column names to meet this requirement.
Pick a field to aggregate values for specified group & column	Click in this field to select the numeric result field to use in aggregate calculations for the selected subgroup. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
How would you want to aggregate values?	Select the aggregate function you want to apply to subgroups from the list: ■ `Average` ■ `Count` ■ `Maximum` ■ `Minimum` ■ `Sum`

	Or use the Path Selector to supply the function name dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Analyze Datasets with RAQL Queries

You can use the **RAQL** block to query and analyze large datasets using MashZone NextGen Analytics and the Real-Time Analytics Query Language (RAQL). RAQL allows you to work with large datasets to perform analysis and combine that with other information or transform it in a mashup.

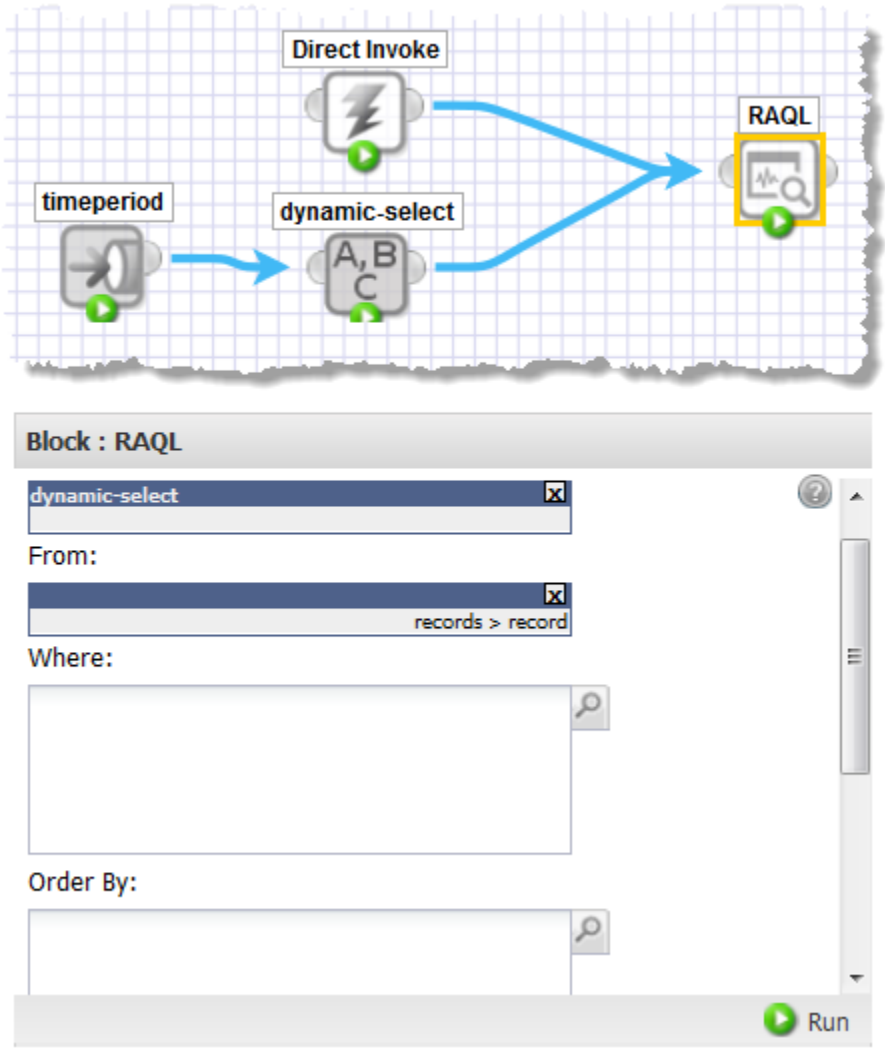
Note: This block supports many, but not all, aspects of RAQL. You cannot load data from a file, use snapshots as a dataset or load datasets from the MashZone NextGen Analytics In-Memory Stores. See the **From** property (below) for more information on the types of datasets and access mechanisms that you can use with this block.

Select	Use this property to define column selections and optional partition or windows in the Select and Over clauses for a RAQL query. You can also apply analytic functions in the Select clause if the query includes an Over or Group By clause. See "Select Techniques" on page 705 and "Over Techniques" on page 705 for a list of syntax options. See "Built-In RAQL Functions" on page 775 for information on the built-in RAQL functions. Or click to open the Path Selector list and provide this clause dynamically from an input parameter or other result. See "Dynamic RAQL Queries in Wires " on page 344 and "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
From	This clause is automatically completed when you connect the results of another block as the dataset input for **RAQL**. The results you connect as input determine whether the block uses streaming to load data. Streaming helps performance with large datasets. If you connect: ■ **DirectInvoke**, the block uses streaming. ■ **SQL**, the block uses streaming. ■ A mashable, mashup or *any* other Wires block, this block does *not* use streaming.

	In addition, the results that you connect as input must be a complex document with repeating items that fit the RAQL data model. See "Structure, Format and Access to Datasets with RAQL" on page 691 for more information.
Where	Use this property to define filter conditions in a Where clause for a RAQL query. See "Where Techniques" on page 707 for a list of syntax options. See "Built-In RAQL Functions" on page 775 for information on the built-in RAQL functions. Or click to open the Path Selector list and provide this clause dynamically from an input parameter or other result. See "Dynamic RAQL Queries in Wires " on page 344 and "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Order By	Use this property to define sort criteria in an Order By clause for a RAQL query. See "Order By Techniques" on page 708 for a list of syntax options. See "Built-In RAQL Functions" on page 775 for information on the built-in RAQL functions. Or click to open the Path Selector list and provide this clause dynamically from an input parameter or other result. See "Dynamic RAQL Queries in Wires " on page 344 and "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Group By	Use this property to define grouping and aggregation criteria in a Group By clause for a RAQL query. See "Group By Techniques" on page 707 for a list of syntax options. See "Built-In RAQL Functions" on page 775 for information on the built-in RAQL functions. Or click to open the Path Selector list and provide this clause dynamically from an input parameter or other result. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Dynamic RAQL Queries in Wires

You can build any clause for a RAQL query dynamically using input parameters and the *String Builder* block, such as this example:

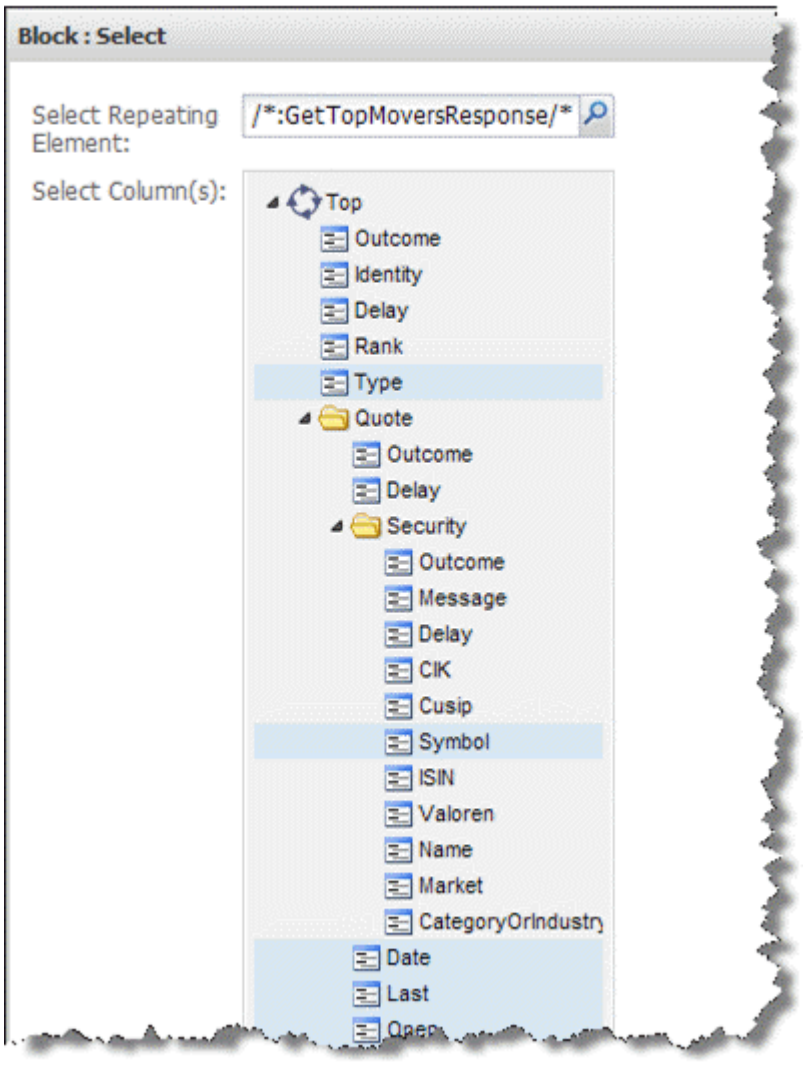


In this example, the Select clause is built dynamically in **String Builder** using an input parameter and the Path Selector list to pull in and combine dynamic content with the literal text of the clause. This is then assigned to **Select** in **RAQL** using the Path Selector.

Select Fields

The **Select** action allows you to select specific fields, and omit others, for all repeating items from a result. For example, you can select just the title and link for all news articles in an RSS feed. Connect a block with document-type results to this action.

Select Repeating Element	Click in this field to select the repeating items ( ) whose fields you want select and omit. See "Select Fields or Paths for Block

	Properties with the Path Selector" on page 372 for more information.
Select Column(s)	Click to see a tree of all the fields within the selected repeating items. Expand this as needed. Then select one or more fields within this structure that contain data that you want to include in the result. All other fields are omitted. 

Add an Item to a SharePoint List

The *Add Item toSharePoint List* block allows you to add new items to an existing SharePoint List and get either the results of the full updated list, including the newly added item, or detailed results of the item just added.

Connection	Select the connection for the SharePoint server with the site and list that you want to update. **Note:** If no SharePoint connections have been configured in MashZone NextGen, this property shows a message and you cannot use this block until a connection to SharePoint has been configured by a MashZone NextGen administrator. If the SharePoint server you need to work with is not in this list, contact your MashZone NextGen administrator.
User Credentials	This section is collapsed by default. It defines the credentials (user account and permissions) used to connect to this SharePoint server and sites. Use the ▼ and ▲ buttons to expand or collapse this section. The permissions for the credentials you use determine which SharePoint sites and lists you can use in this block. Based on configuration, this may default to the credentials defined for the connection you have selected, also known as the *service account*. Or you may be required to supply your credentials for this SharePoint server. **Note:** Some SharePoint server connections may disable the use of user credentials based on configuration. In this case, this section only displays credential information. To use your own SharePoint credentials to connect to this server, change or enter this account information and click **Save**. **Important:** Updating these credentials creates or updates *your* MashZone NextGen credentials for this SharePoint server (which are saved as MashZone NextGen User Attributes). This can affect other mashups or mashables that work with this SharePoint server if they use your credentials rather than the service account. To remove your credentials from this block, click **Remove**. Depending on configuration for this server, the service account credentials may default. This also *removes* your MashZone NextGen User Attributes for this SharePoint server.
Site Name	Once you have the connection and user credentials set, choose the SharePoint site with the list you want to update.

List Name	Choose the SharePoint List you want to update.
Return	Choose the scope of the results for this block: ■ **Entire list**: this is the default, which returns the results for this list including the newly added item. ■ **Only the added item**: set this option if you need a simple confirmation or more detailed information for the item just added to this list.
column-name	Once you have identified the server, site and list that you want to update, MashZone NextGen retrieves information on the columns in this SharePoint List and adds a property to the block for each column. Use these properties to set the value of the item to add to this List.

Run a SQL Statement

Use the **SQL** block to run a SQL statement for a database that has been configured by your MashZone NextGen administrator. You define the SQL statement that the block should run.

This block can be used as an information source by running a SQL query. You can also use it to run stored procedures or to perform insert, update or delete operations.

> Your MashZone NextGen administrator must define a *datasource* (database connection information) before you can use this block.
>
> MashZone NextGen administrators can also configure MashZone NextGen to reject insert, update or delete operations for a database.

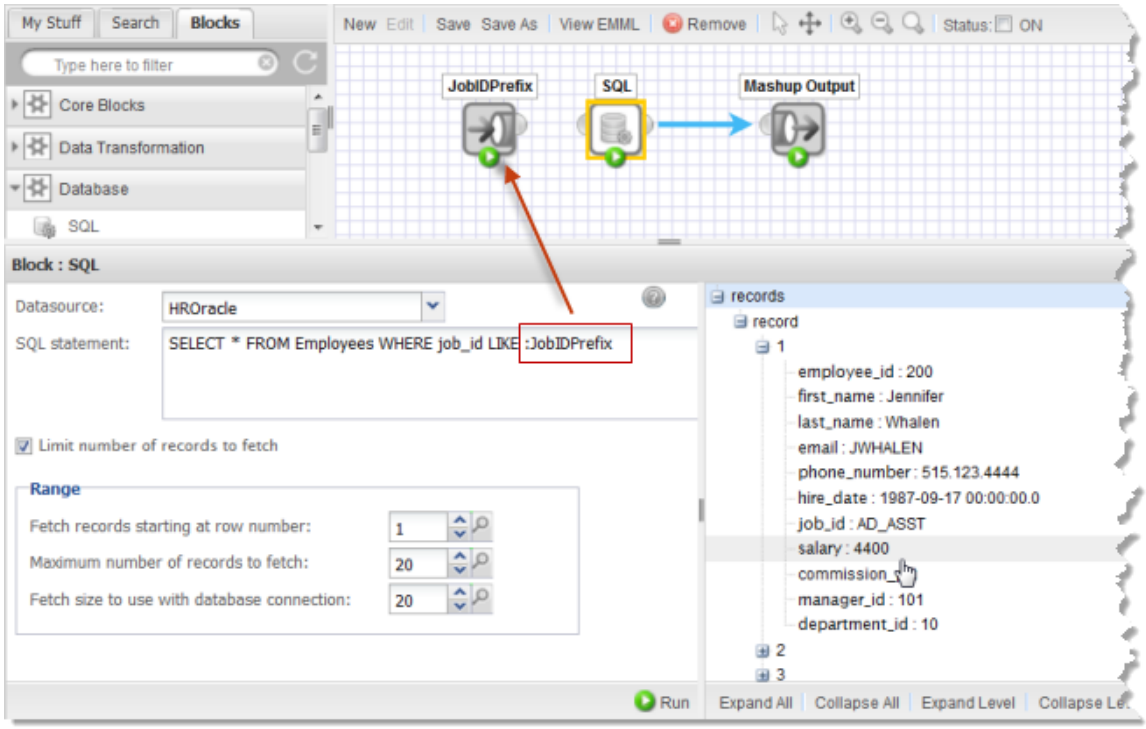
Datasource Name	Select the datasource for the database you want to work with. If there is no datasource for this database, contact your MashZone NextGen administrator to add one.
Enter SQL Statement	Specify the SQL statement for this block. Either: ■ Enter the exact SQL statement to run. You can safely include dynamic values for query parameters in this SQL statement using the `:input-param-name` syntax. See "Using Input

	Parameters as SQL Query Parameters" on page 350 for more information and examples. ■ Click to open the Path Selector list and select an input parameter or other node to supply the SQL statement dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. **Important** Supplying part of a SQL statement through the Path Selector list or using other blocks, such as String Builder, has some risks as this can allow a type of Internet attack known as SQL Injection. For more information, see the "<sql>" on page 492 topic.
Statement Type	If the SQL statement is provided dynamically, from an input parameter or other block, set the appropriate option to indicate whether this statement is a **Query statement** or an **Update statement** (for insert, update or delete operations).
Limit the number of records to fetch	Set this option if you want to limit the results from this SQL query or stored procedure to a specific set of rows. This is set by default to fetch the first 20 rows in Wires. This is a good practice in general to help ensure good performance for the mashup.
Fetch records starting at row number	If you have chosen to limit the records to fetch, select or enter the number of the first record to return. Or click to open the Path Selector list and provide this number dynamically from an input parameter or other result. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Maximum number of records to fetch	If you have chosen to limit the records to fetch, select or enter the total number of rows to return. Or click to open the Path Selector list and provide this number dynamically from an input parameter or other result. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Fetch size to use with database connection	Select or enter the number of records that this mashup should request from this datasource at one time. Or click to open the Path Selector list and provide this number dynamically from an input parameter or other result. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

	Setting a fetch size is generally a good idea when the dataset for this query may be large. It helps to optimize how the mashup retrieves the results, both to improve performance and ensure adequate memory for the query. For more information, see the <sql> topic.

Using Input Parameters as SQL Query Parameters

You can insert the value of an Input block as a value for a condition in a SQL statement by typing in :*input-block-name* as shown in this example:



Note: Using this syntax to supply input parameters to a SQL query removes any risk of an Internet attack known as SQL Injection.

In this example, the parameter value is a string that must include the % symbol for the query to work properly:

To use input parameters in a SQL statement:

1. You *must* add the **Input** block(s) to the mashup before you enter the SQL statement. See "Add Input Parameters" on page 327 for more information.
2. Change the Input block names to something meaningful, if needed. See "Changing Mashup Input Parameter Names" on page 374 for more information.
3. Then enter the SQL statement in the *Enter SQL Statement* property using the input parameter names you have assigned.

Build Strings

The **String Builder** action allows you to build a string value dynamically. You can combine any number of single values from these sources:

- The results of other blocks
- Global or user attributes
- Input parameters
- Literal text.

String Fragments	This contains one or more properties where you can: ■ Enter literal text. ■ Click to open the Path Selector list to supply a string fragment dynamically. Then add an input field, select an input field, select a global or user attribute or select a path to a field from some other block to supply this fragment dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

	Note: Fields that you select from other block results *must* contain a single value. They cannot be fields in a repeating item. The order of the fragments defines the final order of the combined string.
Add More	Click this button to add another fragment to build this string.
Remove All	Click this button to remove all the existing fragments.

Sort Results

The **Sort** action sorts repeating items in results based on a key field and order you choose. Connect a document-type result as the input to **Sort**.

Repeating element to sort	Generally set automatically by Wires to the repeating items in the results from the input. To change this, delete the existing path and use the Path Selector button to find the repeating items you want to sort. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
Fields to sort on	Choose one or more fields in these repeating items to sort by, in the order you want them sorted. For each field, complete the fields for one sort criteria. Click the **+** button to add another sort criteria
left field in sort criteria	Use the Path Selector button to find a field to sort results by. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
or	Click this button to change the sort order for this field. Defaults to ascending.
datatype	Choose the datatype of the data for this field: ■ `Date` = for dates, times or dates and times ■ `Numeric` = for numbers ■ `String` = for text values

MM/dd/yyyy	If the field is a date or time, select the pattern that represents the format of values in this field. See "Date/Time Format Patterns" on page 353 for information on the meaning of the characters used in date/time patterns.
Case Sensitive	Makes text sorting case sensitive.

Date/Time Format Patterns

The following characters are used in date/time format patterns:

Pattern	Usage
`a`	AM or PM for 12-hour times
`dd`	2-digit day of the month, such as 05
`EEE`	3-character abbreviation of the day of the week, such as Thu
`EEEE`	full name of the day of the week, such as Thursday
`HH`	2-digit hour in a 24 hour clock (0-23)
`hh`	2-digit hour in a 12 hour clock (01-12)
`mm`	2-digit minutes
`MM`	2-digit month, such as 12
`MMM`	3-character month abbreviation, such as Dec
`MMMM`	Full month by name, such as December
`ss`	2-digit seconds
`sss`	3-digit milliseconds
`yyyy`	4-digit year, such as 2005
`z`	Timezone

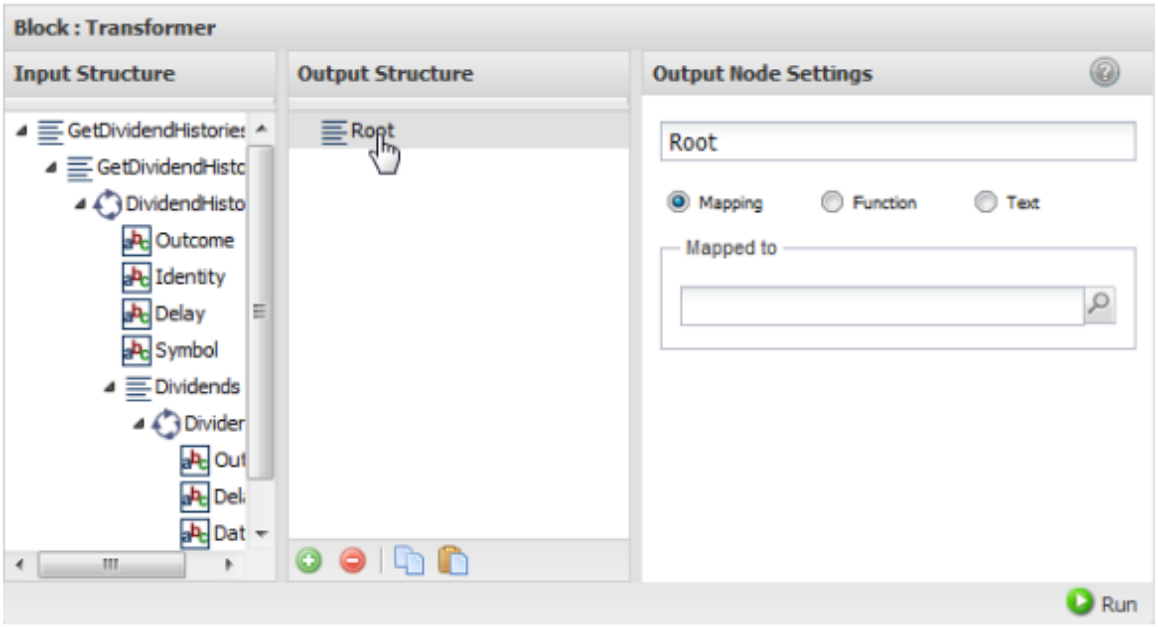
Transform Results

The **Transformer** action allows you to change the structure and data from document results to a document structure you define.

To define the transformation

1. Connect document-type results from another block as input to **Transformer**.

   The Block Properties for this action displays three panes with the structure of the input document, a pane where you build the output structure, and a pane with properties you can assign to the currently selected output node or field.



Tip: It helps to maximize the Block Properties/Preview panel with the button while you work with both structures.

Both the input and output structure can contain:

- = a root or structure node that contains children but does not repeat. The root node completely wraps the entire document of the results.
- = a repeating node, usually with children.

- = a field (node) that contains data. If the node name begins with `@` this is an XML attribute. Otherwise it is an XML element.

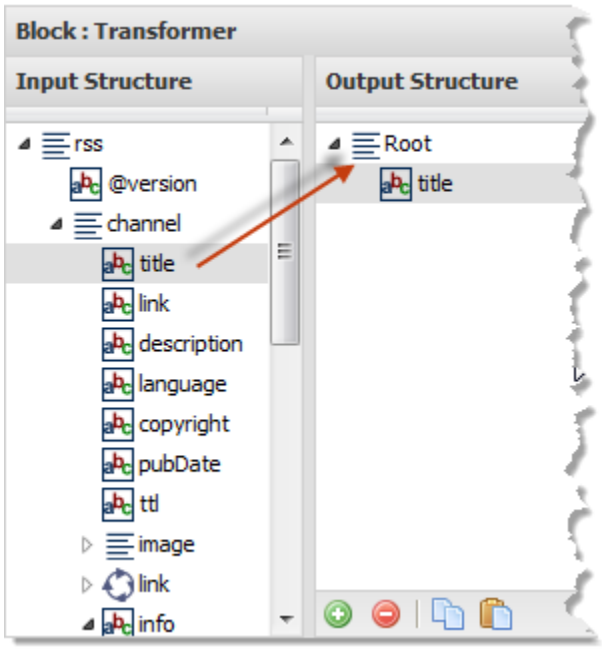
2. Change the name for the root node, if desired, in the Ouptut Node Settings field.
3. Build the output structure under the root node and define what data should populate the results. To build the output structure and data:
   - Drag Input Nodes to Add Nodes to the Output: you can drag individual fields, entire structures or repeating nodes. You can then delete specific nodes from the output, rename output nodes or use node properties to transform the output data.
   - Manually Add Fields or Structure Nodes for more control over the structure and order. Then use node properties to populate data.
   - Map Input Nodes to the Output to populate data in the output structure.
   - Enter or Calculate Data for output fields. You can enter literal data or use functions to calculate data.
   - *Delete output nodes*: select the output node and click .
   - *Rename nodes*: in the Output Node Settings pane.

     Node names must be valid XML names, starting with a letter and containing only letters, numbers, periods(.), underscores (_) or dashes (-).
   - *Copy/paste node property settings* from one output node to another with the and buttons in the Output Structure toolbar.
4. Use **Run** to preview the results as needed.

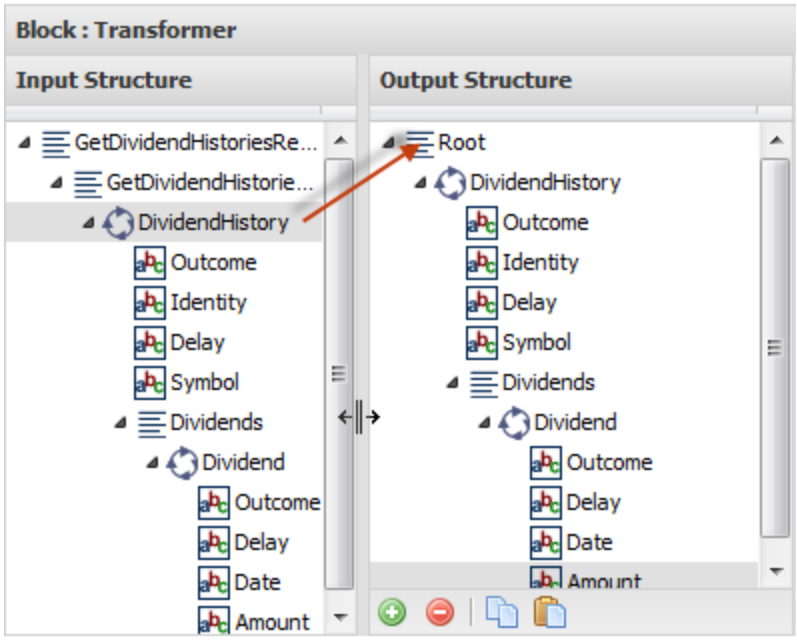
Drag Input Nodes to Add Nodes to the Output

You can drag nodes from the input structure to add them and automatically map the input data to the output. You can also rename these nodes or use functions to transform the data.

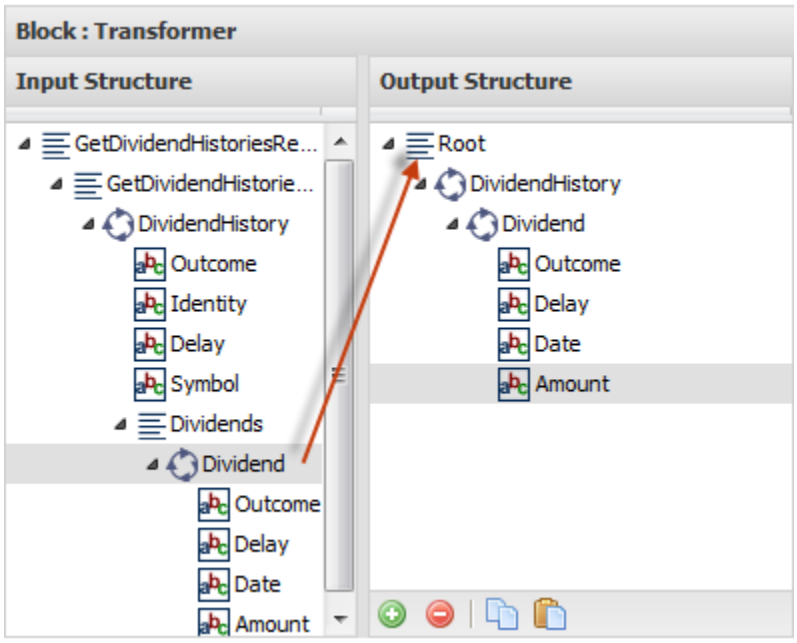
1. Select a field, structure or repeating node from the input.
2. Drag it into the output structure and drop on top of the output node that should contain this new node or structure. Nodes are always added at the end of any existing children nodes.
   - If you selected a data field, that single field is added to the output and automatically mapped to the input.



- If you selected a structure node, the entire structure is copied to the output and automatically mapped.
- If you selected a repeating node, the entire structure of that repeating nodes is copied to the output as a repeating node and automatically mapped.



If the input you selected is itself inside a repeating node, then all levels of repeating nodes are copied.



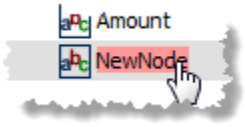
You can then rename, delete or change mapping or data transformation properties for any or all of the new output nodes.

Manually Add Fields or Structure Nodes

In some cases, dragging doesn't give you the order or exact structure that you want. You can also use the toolbar to add nodes one at a time and then manually map them as needed.

> **Note:** You cannot manually add repeating nodes to the structure. Instead, drag a repeating input structure and modify the copied nodes.

1. Select the output node that should contain the new node.
2. Click ⊕ to add a new, unmapped node.



3. To change the default name for the new node, double-click the node and enter a new name.

   To make the new node an attribute of its parent, start the name with `@`, such as `@id`.

4. Either:

   - Add new nodes under this, manually or using drag-and-drop, to make this a structure node.

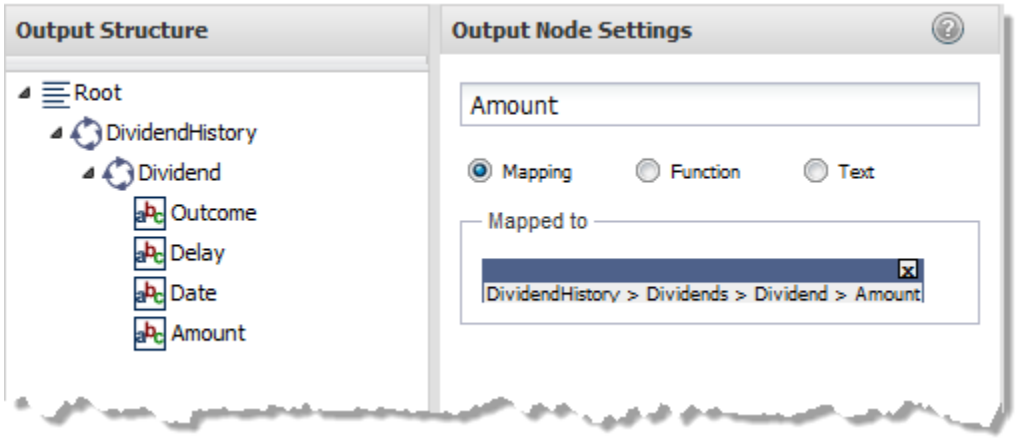
- "Map Input Nodes to the Output" on page 358 or "Enter or Calculate Data" on page 359 for the node if it should be a data field.

Map Input Nodes to the Output

Mapping defines how data from the input document or other blocks populates the result of **Transform**. You can map data fields that you have manually added to the output structure or change mapping information automatically assigned from drag-and-drop additions.

1. Select the output data field you want to map.
2. Set the **Mapping** option in the Output Node Settings panel.
3. Click to open the **Path Selector** list and select:
   - A data field from the input document.
   - A data field from the results of any other block in the mashup.
   - An existing input field for the mashup.
   - Add a new input field.

Once mapped, the node properties show the mapping path:



You can map single fields to single fields or repeating fields in the output. Similarly, you can map repeating fields to single fields or repeating fields in the output. You can also map entire documents to output fields. The result will be:

From > To	Result
>	Single input value populates the single output field.
>	Single input value populates all the repeating output fields.

From > To	Result
>	The values from all repeating input fields are joined, in order, with spaces between the values and this single joined value populates the output field.
>	Each repeating field populates a single corresponding output field.
Document > or	The full structure of the input document is appended as a child of the single or all repeating output fields.

Enter or Calculate Data

In addition to mapping, you can enter literal values for output fields or use functions to calculate or transform data.

1. Select the output data field you want to map.
2. To provide a literal value, set the **Text** option in the Output Node Settings panel and enter the value in the Text field.
3. To use a function to calculate a value, set the **Function** option in the Output Node Settings panel and:
   - Select the function from the list. MashZone NextGen has a set of built-in functions, but MashZone NextGen developers or administrators can add functions to this list.
   - Complete the properties for the function. See "Built-in Functions for Decorator, Mapper and Transformer Blocks" on page 359 for property information for the built-in functions.

Built-in Functions for Decorator, Mapper and Transformer Blocks

The **DataDecorator**, **Mapper** and **Transformer** actions in Wires provide a set of built-in functions that you can use to transform the data output from these blocks. MashZone NextGen developers may also add additional functions from the XPath standard or custom XPath functions for your use.

The built-in functions you can use include:

String	Number	Date or Boolean
Concatenate	Absolute	Add to Date
Encode URI	Add	Date Formatter

String	Number	Date or Boolean
Join With	Ceiling	Date from Days
String Replace	Currency Value	Date from Seconds
String Matches	Distance Between Geo Points	Subtract from Date
Substring After	Divide	To Boolean
Substring Before	Floor	To Date
Substring	Set Decimal Places	To Datetime
To Camel-Case	Multiply	
To Lower-case	Round	
To String	Subtract	
To Title-Case	To Decimal	
To Upper-case	To Integer	
	To Number	

Absolute

Returns the absolute (positive) value for a numeric field that you map to this output field. If you have not already mapped this output field, click to open the **Path Selector**.

Add

Returns the sum of any number of single numbers and single numeric fields.

This function can also add any number of single numbers to each instance of a repeating numeric field *only when* the repeating numeric field is within an input to the current block. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information on selecting fields from block inputs.

You can:

- Enter a literal number in any **Value *n*** field.

- Click to open the **Path Selector** list in any **Value *n*** field. Map this to an input field, to any single field with numeric data from block results in this mashup or to a repeating numeric field that is an input to the current block.
- Click **Add another Value** to add fields for more numbers to include in this calculation.

Add to Date

Updates a single datetime input field by adding a specified period of time. This function can also update each instance of a repeating datetime field.

Note: You cannot directly use an input parameter as the input to **DataDecorator** or **Transformer**. To update a datetime input parameter, wrap the input parameter in a node using the **Document** block and use this as the input to **DataDecorator** or **Transformer**.

To specify the time period to add, enter an integer or use the **Path Selector** to set the number dynamically. Then choose the period:

- Years
- Months
- Days
- Hours
- Minutes
- Seconds

Ceiling

Returns the next highest integer value for a numeric field that you map to this output field. If you have not already mapped this output field, click to open the **Path Selector**.

Concatenate

Joins two or more text fields or literals into a single string to use as data for the current field. You can either:

- Enter a literal value in any **String *n*** field.
- Click to open the **Path Selector** list in any **String *n*** field. Map this to an input field or any field from block results in this mashup.
- Click **Add another String** to add fields for more strings.

Currency Value

Converts a fully-formatted currency amount (a string) to a numeric value that can be used in calculations. This also supports currency values in exponential forms such as `$1.6E8`.

- If you have not already mapped a field to the **Extract number from currency value** property, enter a formatted currency amount or click to open the **Path Selector** and map a field.
- Select the symbol (`Period separated fractions` or `Comma separated fractions`) that is used as the decimal symbol in this currency field.

The result of this function *always* uses a period as the decimal symbol. No currency or thousand separator symbols are included in the result. Plus or negative signs are retained.

Date Formatter

Changes the format of a date, time or date and time field that you map to this output field. If you have not already mapped this output field, click in the **Select a Date** property to open the **Path Selector** and map a field.

Then define the format of the input date and/or time and the output format you want for this date and/or time. For both properties, you can:

- Select an existing format string.
- Enter a format string directly.
- Or click to open the **Path Selector** to supply the formats from input fields or fields from any block results in this mashup.

Format strings for dates and times use letters and symbols in a pattern indicating what each character in the date or time represents. The most common patterns for format strings include:

Pattern	Usage
`a`	AM or PM for 12-hour times
`dd`	2-digit day of the month, such as 05
`EEE`	3-character abbreviation of the day of the week, such as Thu
`EEEE`	full name of the day of the week, such as Thursday
`HH`	2-digit hour in a 24 hour clock (0-23)
`hh`	2-digit hour in a 12 hour clock (01-12)
`mm`	2-digit minutes
`MM`	2-digit month, such as 12

Pattern	Usage
`MMM`	3-character month abbreviation, such as Dec
`MMMM`	Full month by name, such as December
`ss`	2-digit seconds
`sss`	3-digit milliseconds
`yyyy`	4-digit year, such as 2005
`z`	Timezone

You can combine this with `/`, `-` or `.` as delimiters between portions of the date. Times always use `:` and `.` delimiters. For example:

```
01/29/2010 01:15:01.45AM uses MM/DD/YYYY hh:mm:ss.Sa
31-03-2001 uses DD-MM-YYY
21:00 uses HH:mm
```

The complete list of characters and symbols that are valid in patterns are defined in the `java.text.SimpleDateFormat` class. For detailed information, see Java API documentation for the JDK version used in your environment.

Date from Days

Returns the value of the date in this field as an integer count of the number of days to this date from January 1, 1970. This is one common representation of dates for some software.

Date from Seconds

Returns the value of the date in this field as an integer count of the number of seconds to this date from January 1, 1970. This is one common representation of dates for some software.

Distance Between Geo Points

Returns the distance between two locations that are identified by lattitude and longitude.

Enter the lattitude and longitude of both locations or click to open the **Path Selector** to find the fields with this information.

Divide

Returns the result of dividing one number or numeric field from another number or numeric field.

This function can also divide each number of a repeating numeric field by a single number (or vice versa) *only when* the repeating numeric field is within an input to the current block. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information on selecting fields from block inputs.

Enter a literal number or click to open the **Path Selector** list in either **Divide** or **By** to supply the numbers for this calculation.

Encode URI

Returns the text of the field that you map to this output field encoded as a URI. If you have not already mapped this output field, click to open the **Path Selector**.

Floor

Returns the next lowest integer value for a numeric field that you map to this output field. If you have not already mapped this output field, click to open the **Path Selector**.

Join With

Joins two or more text fields or literals, separated by a delimiter such as a comma, into a single string. You must supply the delimiter character(s) and at least two fields or literal strings to join.

For the delimiter, enter a literal set of characters or click to open the **Path Selector** and map this property to an input field or a field from any block results in this mashup.

For the strings to join:

- Enter a literal value in any **String *n*** field.
- Click to open the **Path Selector** list in any **String *n*** field. Map this to a new input field, an existing input field or a field in the input document or any document present in this mashup.
- Click **Add another String** to add fields for more strings.

Multiply

Returns the product of any number of single numbers and numeric fields.

This function can also multiple any number of single numbers with each instance of a repeating numeric field *only when* the repeating numeric field is within an input to the current block. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information on selecting fields from block inputs.

You can:

- Enter a literal number in any **String *n*** field.
- Click to open the **Path Selector** list in any **String *n*** field. Map this to an input field, to any single field with numeric data from block results in this mashup or to a repeating numeric field in the input to the current block.

- Click **Add another Field** to add fields for more numbers to include in this calculation.

Not

Returns the opposite of "To Boolean" on page 367 based on this nodes value or content.

Round

Rounds to an integer a numeric field that you map to this output field. If you have not already mapped this output field, click to open the **Path Selector**.

Set Decimal Places

Sets the number of decimal places for a numeric field and rounds the result if needed.

If you have not already mapped a field to the **Formatted value of** property, you can enter a number or click to open the **Path Selector** list and map a numeric field.

Enter the number of decimals places to enforce in the result or click to open the **Path Selector** and supply this number dynamically from an input field or the results of another block.

The number of decimal places can be zero or positive integer. If the numeric value(s) from this field have more decimal places, the result is rounded to fit.

String Matches

Returns `true` if the characters or value you identify are found in another field.

The **String Fragment in** property is mapped, by default, to the field you apply this function to. Delete this mapping, if desired, and enter a literal value or click to open the **Path Selector** and map this property to an input parameter or any text field in block results in this mashup.

Enter one or more literal characters to search for a matching fragment in the **Matches With** field. You can also click to open the **Path Selector** and map this property to any text field to supply this value.

Note: The search for the fragment is *not* case sensitive.

String Replace

Replace every occurrence of one character or a set of contiguous characters within the text of the field that you map to this output field.

Enter a literal value in the **Replace String in** property. Or click to open the **Path Selector** and map this property to an input field or any text field in block results in this mashup.

Enter one or more literal characters to be replaced in the **Look for Pattern** field. You can also click to open the **Path Selector** and map this property to any text field to supply this value.

Note: Search for the pattern is *not* case sensitive.

Enter one or more literal characters to use as the replacement for found patterns in the **Replace With** field. You can also click to open the **Path Selector** and map this property to any text field to supply this value.

Substring

Returns the specific characters from a literal string or the field that you map to this output field based on the character positions you specify.

Enter a literal value in the **String Fragment of** property. Or click to open the **Path Selector** and map this property to an input field or any text field in block results in this mashup.

Enter the position of the first character to extract, starting with 1, in the **Start At** field. You can also click to open the **Path Selector** and map this property to a numeric field to supply this value.

Optionally, enter the position of the last character to extract in the **End At** field. You can also click to open the **Path Selector** and map this property to a numeric field to supply this value. If you do not specify an ending character, this extracts everything from the start character through the end of this field.

Substring After

Returns all the characters from a literal string or the field that you map to this output field that occur after the first substring found that matches the string or character that you specify. If no matching substring is found, this returns an empty string.

Enter a literal value in the **String Fragment of** property. Or click to open the **Path Selector** and map this property to an input field or any text field in block results in this mashup.

Enter a single literal character to search for in the **After String** field. You can also click to open the **Path Selector** and map this property to a numeric field to supply this value.

Substring Before

Returns all the characters from a literal string or the field that you map to this output field that occur before the first substring found that matches the string or character that you specify. If no matching substring is found, this returns an empty string.

Enter a literal value in the **String Fragment of** property. Or click to open the **Path Selector** and map this property to an input field or any text field in block results in this mashup.

Enter a single literal character to search for in the **Before String** field. You can also click to open the **Path Selector** and map this property to a numeric field to supply this value.

Subtract

Returns the result from subtracting one number or numeric field from another number or numeric field.

This function can also subtract one number from each instance of a repeating numeric field (or vice versa) *only when* the repeating numeric field is within an input to the current block. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information on selecting fields from block inputs.

Enter a literal number or click to open the **Path Selector** list in either **Subtract** or **from** to supply the numbers for this calculation.

Subtract from Date

Updates a single datetime input field by subtracting a specified period of time. This function can also update each instance of a repeating datetime field.

> **Note:** You cannot directly use an input parameter as the input to **DataDecorator** or **Transformer**. To update a datetime input parameter, wrap the input parameter in a node using the **Document** block and use this as the input to **DataDecorator** or **Transformer**.

To specify the time period to add, enter an integer or use the **Path Selector** to set the number dynamically. Then choose the period:

- Years
- Months
- Days
- Hours
- Minutes
- Seconds

To Boolean

Returns `false` if the value or contents of this node are:

- Numeric with zero as a value.
- String and an empty string.
- A structure node with no children (also known as an empty node set).

Any other value or the presence of children will return `true`.

To Camel-Case

Returns the text value for this field with all spaces removed and the first letter of each word as upper-case letters. With an initial value of `The original value.`, for example, this returns `TheOriginalValue.`

See also "To Title-Case" on page 369.

To Date

Returns the text value of the field that you map to this output field converted to a date. Text values *must* be in the format `YYYY-MM-DD`. If the value of this field cannot be converted to a date, this returns as error.

If you have not already mapped this output field, click to open the **Path Selector**.

To Datetime

Returns the text value of the field that you map to this output field converted to a valid date and time combination in the format `YYYY-MM-DDThh:mm:ss.szzzzzz`. If the value of this field cannot be converted to a date or to a date and time, this returns an error.

Enter a literal date or date/time value in the **Select a date** property. Or click to open the **Path Selector** and map this property to an input field or any text field in block results in this mashup.

Choose the format in which the input date or date/time value(s) appear in the **Select input date format** property. Or click to open the **Path Selector** and select a field that contains the appropriate date or date/time format for the input.

To Decimal

Returns the text value of the field that you map to this output field converted to a decimal value. If the value of this field cannot be converted to a number, this returns `NaN` (not a number) as the value.

If you have not already mapped this output field, click to open the **Path Selector**.

To Integer

Returns the text value of the field that you map to this output field converted to an integer value. If the value of this field cannot be converted to a number, this returns `NaN` (not a number) as the value.

If you have not already mapped this output field, click to open the **Path Selector**.

To Lower-case

Returns the text value in all lower-case letters of the string field that you map to this output field. If you have not already mapped this output field, click to open the **Path Selector**.

To Number

Returns the text value of the field that you map to this output field converted to a numeric value. Either integer or decimal values are valid. If the value of this field cannot be converted to a number, this returns `NaN` (not a number) as the value.

If you have not already mapped this output field, click to open the **Path Selector**.

To String

Returns the text value of the field that you map to this output field converted to a string type. If you have not already mapped this output field, click to open the **Path Selector**.

To Title-Case

Returns the text value for this field with the first letter of each word as upper-case letters. With an initial value of `The original value.`, for example, this returns `The Original Value.`

See also "To Camel-Case" on page 368.

To Upper-case

Returns the text value in all upper-case letters of the string field that you map to this output field. If you have not already mapped this output field, click to open the **Path Selector**.

Build URLs

The **URLBuilder** action allows you to create URLs dynamically to use as input to DirectInvoke or other blocks. You can either:

- Copy and Update a URL
- Provide the Complete URL Dynamically
- Build a URL from Components

URL	■ Leave this property blank if you are building the URL from components. ■ To copy and update a URL, paste or enter the example URL in this field. Then click **Run** or press **Tab**. Wires separates the example URL into its various components which you can now update individually, or assign input fields or other paths to provide this data dynamically. ■ To provide a fully dynamic URL, click and add an input field or select a field from some other block. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Protocol	Use this property to update existing URLs or build URLs from components. Select `HTTP` or `HTTPS` as the protocol for the URL.
Domain	Use this property to update existing URLs or build URLs from components. Either: ■ Enter the domain directly. ■ Click to open the Path Selector list. Then add an input field, select an input fiels or select a path to a field from some other block to supply the domain dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. For example: `www.example.com`
Add a new path	Use this property to update existing URLs or build URLs from components. Click to add one *step* of a path to the domain. For example: `http://www.example.com/`*new-path-step* You can add any number of path steps in the order in which they should appear in the final URL.
Paths	You can either: ■ Enter a value for this portion of the path. ■ Click to open the Path Selector list. Then add an input field, select an existing input field or select the path to a field from some other block to supply this portion of the path dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. For example: `http://www.example.com/{$care-giver-type}`
Add a new parameter	Use this property to update existing URLs or build URLs from components. Click to add a parameter to the URL. For example: `http://www.example.com/md?specialty=pediatrician` You can add any number of parameters to the URL.
Parameters	Use this property to update existing URLs or build URLs from components.

	For each parameter, enter the parameter name in the left field. Define the parameter value in the right field. Either: ■ Enter a value. ■ Click to open the Path Selector list. Then add an input field, select an existing input field or select the path to a field from some other block to supply the parameter value dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information. For example: `http://www.example.com/{$care-giver-type}?specialty={$specialty}`

Copy and Update a URL

In the **URL** property, paste or enter an example of the URL you want to update to make it dynamic. Click **Run**. Wires separates the example URL into its various components which you can now update individually.

To make any component of the URL dynamic, use to open the Path Selector list and add or select the field to supply the dynamic value of this component.

Provide the Complete URL Dynamically

In the **URL** property, click to open the Path Selector list and select either an input field or the path to a field from the results of another block that should supply the full URL dynamically.

Build a URL from Components

Select a **Protocol** and enter a value or select an input field or other block field to provide the **Domain** dynamically.

If needed, use the **Add a new path** and **Add a new parameter** buttons to add additional steps or parameters to the URL. For each path, enter a value or select an input field or other block field to provide the path dynamically. For each parameter, enter a parameter name and either enter the parameter value or select an input field or other block field to provide the parameter dynamically.

For example:

- Protocol = `HTTP`
- Domain = `www.example.com`
- Paths = `{$location}` and `md`
- Parameters = `name: {$name}` and `specialty:{$specialty}`

Would build a URL such as this:

```
http://www.example.com/nw/md?name=tay&specialty=pediatrician
```

Select Fields or Paths for Block Properties with the Path Selector

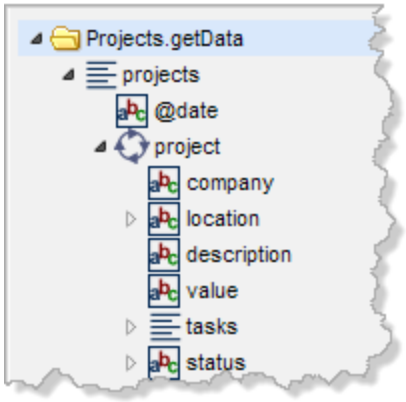
Many block properties must identify a specific field, a structure or a set of repeating items within results from another block. For example, join conditions for the **Join** block identify a field in each result that is being joined that should be compared to determine exactly what data to join.

Other block properties may accept values that you enter, but you can also supply values dynamically when the mashup is run from an input field for the mashup or from a field in the result of another block.

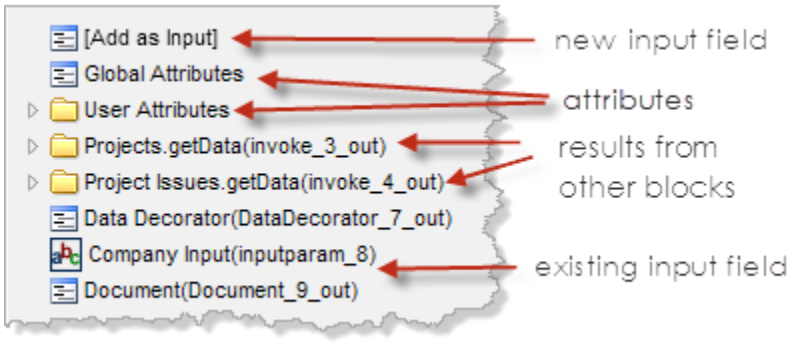
You can easily identify specific fields, structures, or repeating items with the **Path Selector** list. You can also select global attributes, user attributes or existing input fields or add a new input to provide the value for a property using the **Path Selector** list.

- To open the **Path Selector** list:
  - For properties that require a path to a field, structure or set of repeating items, simply click in the property.
  - For properties that allow you to enter values or a path, click .

  The list shows different choices depending on your current block:
  - Just the results from the blocks that are an input to your current block:

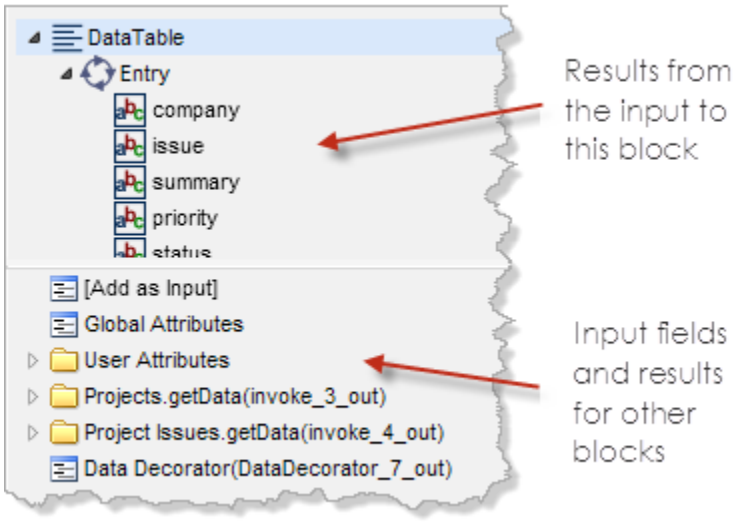


  - Existing input fields and attributes plus the results from any block in the mashup:



You can also add new input fields from the list in this case.

- Results from blocks that are input to this block plus inputs or results from other blocks:



Within results, these icons indicate what kind of node you are selecting:

	Root node
	Non-repeating structure node, containing children
	Repeating node, either a structure with children or a field
	A field with boolean data
	A field with date or time data

	A field with an enumerated list of valid values
	A field with numeric data
	A field with text (string) data

- Click on the field, existing input field, structure or repeating item you want to use for this property.

  The property fills with a path, such as `*:ResultSet/*:Result/*:Title`.

  Each slash indicates a new step down the path of the structure followed by the name of the node at that point in the path. The asterisks and colon (`*:`) before each node name indicate that the namespace, if any, is not used. See "What is a namespace?" on page 73 for more information on namespaces.

- Or click **Add input field** to add a new → **Input** block and connect it to this property.

Use MashZone NextGen Global or User Attributes in Block Properties

You can use MashZone NextGen global attributes or user attributes to fill in block properties for inputs to mashables or mashups. Global MashZone NextGen attributes are meant to be used by any user. User MashZone NextGen attributes have values that are specific to you or to the user running your mashup.

Your MashZone NextGen administrator defines MashZone NextGen global attributes for values that many or all users might need to use as input for a mashable or mashup. You define user attributes in your MashZone NextGen profile. For example, MashZone NextGen attributes can define the username or password to work with a mashable source or a key that is needed to allow Google maps to be used as a view for apps.

To use a MashZone NextGen attribute in a block property, open the *Path Selector* list and select the attribute from the *Global* or *User* folder.

Changing Mashup Input Parameter Names

When you add input parameters to a mashup, using the → **Input** action block, Wires automatically assigns a unique, valid name to the input parameter in one of two forms:

- `inputparam_n`
- *`info-source-param-name_n`*

Where *n* is a unique number and *info-source-param-name* is the name of the information source input parameter that this mashup input parameter is connected to. The label for the input parameter is the same as its name.

Note: In previous releases, you could change input parameter labels, but not their names.

You can change the names of mashup input parameters *only* if:

- The mashup has never before been turned on.

  Important: *Never* change the name of a mashup input parameter once you have turned a mashup on. This can cause apps or other mashups that use this mashup to fail.

- The new names are Valid Input Parameter Names.

Valid Input Parameter Names

To be valid, input parameter names must:

- Be unique within the mashup.
- Not use reserved names from EMML, Java, JavaScript or SQL. Using reserved names can cause the mashup to fail.

  Tip: The easiest way to ensure that you do not use a reserved name is to *always* make your input parameters specific. This also makes them easier for other user to understand.

  For example, use `projectDocument` rather than `document` or `itemCount` rather than `count`. You can also use prefixes or suffixes, such as `myCount` or `client2`.

- Contain only ASCII letters , numbers and the underscore (_) character. If you include spaces in parameter names, they are automatically converted to underscores (_). No other punctuation characters or symbols are allowed in parameter names.

  Note: Names are case sensitive, so `items` and `Items` are different input parameters. It is *not* a good practice, however, to have input parameter names that are very similar.

- Start with a letter. `6sigma` is not a valid input parameter name.

Add If/Else Conditions

One very common requirement for mashups, is to define different processing steps based on conditions, commonly known as *if/else conditions*. In most programming languages, the structure for if/else conditions looks something like this:

```
//one condition to check, if true, do steps in subordinate branch
If A = B
//subordinate branch
{do A
do B
do C}
```

```
//if no conditions are met, do steps in subordinate branch
Else
//subordinate branch
{do B
do D}
```

Wires does not have action blocks that represent If or Else. Instead, you define if/else behavior in a mashup in Wires using *execute conditions* in the advanced properties for blocks. See "If/Else Examples" on page 378 for examples.

You define the condition to be met on the first block in each conditional branch, as an *execution condition*. See "Set Execution Conditions" on page 376 for the basic steps to define an execution condition on a block. Conceptually, the logic would look more like this:

```
//conditional branch
{do A and all steps in this branch if L = M
do B
do C}
//conditional branch
{do D and all steps in this branch if L != M
do E}
```

You should also be aware of "Add If/Else Conditions" on page 375.

Set Execution Conditions

Execution conditions can be set on blocks for any mashable or mashup and for most of the built-in MashZone NextGen built-in action blocks. Custom blocks may also support execution conditions.

To set an execution condition:

1. Select the first block of a branch on the canvas that you want to make conditional.
2. Click the Advanced Properties tab in the block properties pane.
3. Click in the left field of a condition to select the field, input parameter or MashZone NextGen global or user attribute to test in this condition. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
4. If needed, change the comparison function for the condition to:
   - `Contains` = checks to see that the text characters you are testing for are present somewhere in the field you have chosen to test. The comparison is not case sensitive, unless you set the Case Sensitive option.
   - `Does not contain` = checks to see that the text characters you are testing for are *not* present somewhere in the field you have chosen to test. The comparison is not case sensitive, unless you set the Case Sensitive option.
   - `Matches` = checks to see that the characters you are testing for are present somewhere in the field you have chosen to test.
   - `Does not match` = checks to see that the characters you are testing for are *not* present somewhere in the field you have chosen to test.

- `= (equals), != (does not equal), > (is greater than), < ()is less than, >= (is greater than or equals) or <= ()is less than or equals` = compares the entire value of the field you select with the value you set for the condition using standard numeric comparisons.

5. Enter a value to test for this condition in the right condition field, or click to set the value for the header dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.
6. Mark this test as *Case Sensitive*, if needed.
7. Click **Add Condition** to create additional conditions to be met and complete conditions fields as needed.

   Use the top field to indicate whether *all conditions must be met* before the block should execute or *meeting any condition* is sufficient.

> **Note:** MashZone NextGen developers who know XPath well can use the *Advanced Mode* button to customize the XPath expressions for conditions. Use the advance mode with care, however, as it disables the display of conditions in basic mode. If you return the condition to basic mode, your customized XPath expression for the condition is lost.
>
> See "Edit Mashup XPath Expressions in Advanced Mode" on page 391 for more information.

Execution Condition Limitations

There are some specific limitations you must consider when constructing if/else logic in Wires:

- If you want only one condition to be met, *all* of the conditions must be mutually exclusive. The common usage for if/else logic in programming languages is to check each condition until one is found to be true. Once a condition is met, that clause is executed and all subsequent conditions are skipped.

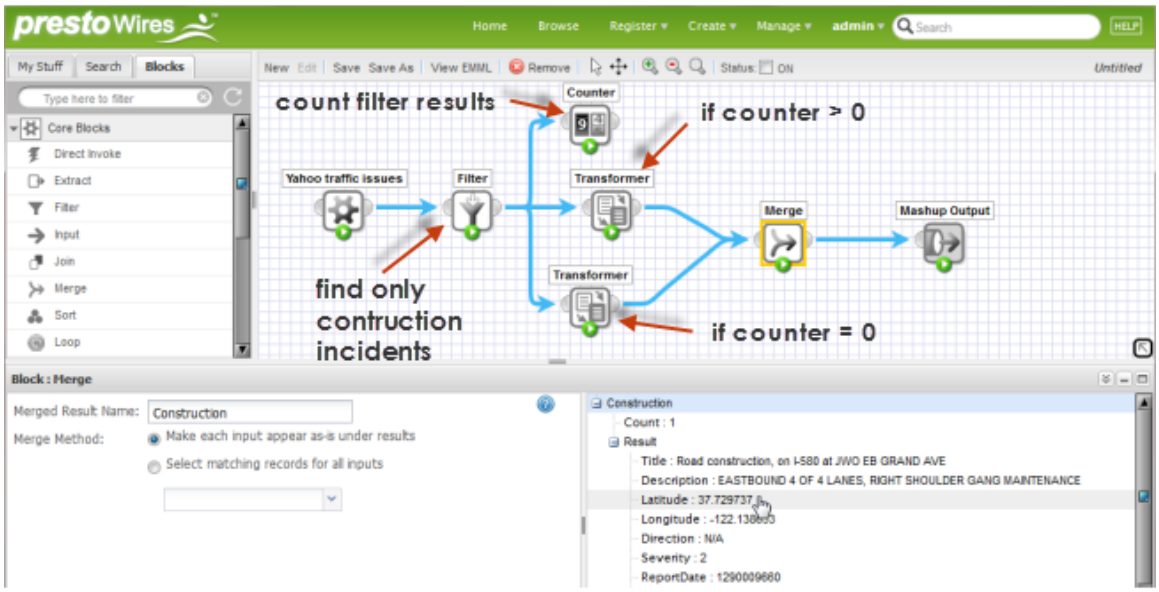
  With execution conditions in Wires, each branch is independent. *Every* branch is evaluated to determine if it should execute without regard to any other branch. If the conditions are not mutually exclusive, multiple branches will run.

- The mathematical operators, such as > or `!=` are best used with numeric data . You may need to use casting functions, such as ToNumber, in the Data Decorator, Mapper or Transformer blocks to make sure that data is treated as numeric.

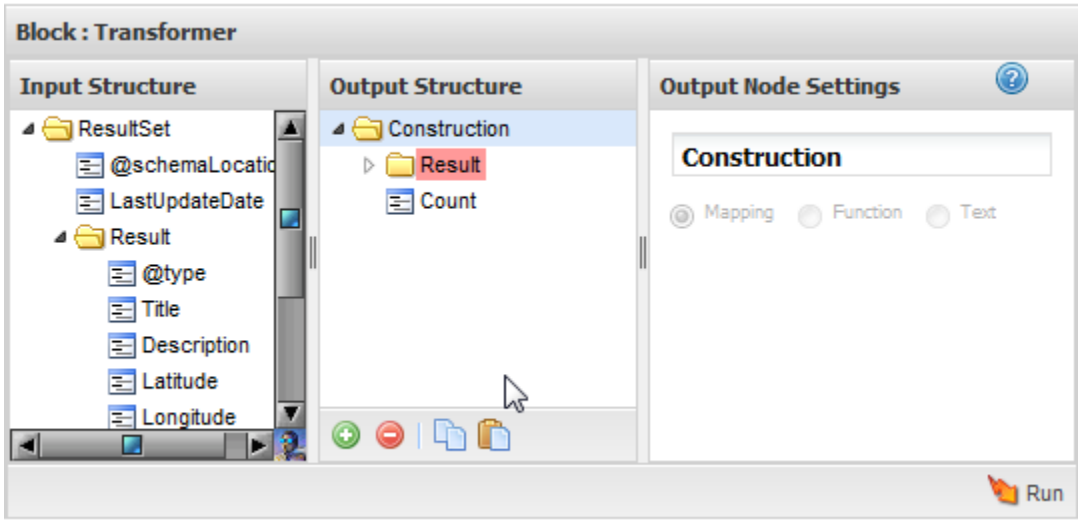
- Block conditions do *not* have a way to specify Else logic (no previous conditions are met). To handle the behavior you want if no conditions are met, you must define a condition that is the reverse of all other conditions. See "If/Else Examples" on page 378 for an example.

If/Else Examples

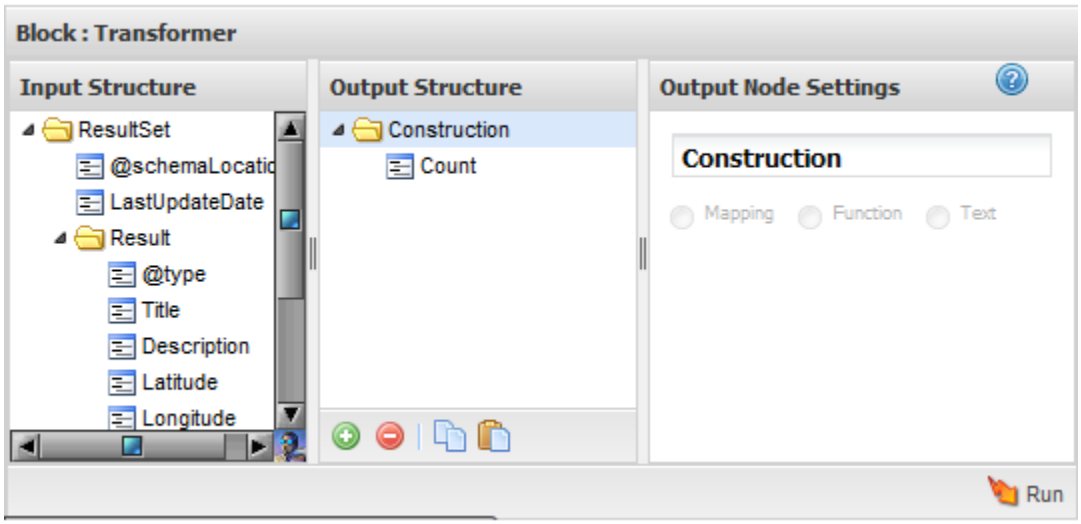
This first example shows a mashup with very simple if/else logic based on the incident type for traffic issues:



The "if clause" consists of a single Transformer block that executes if there is at least one issue from construction and builds the mashup result from those issues.

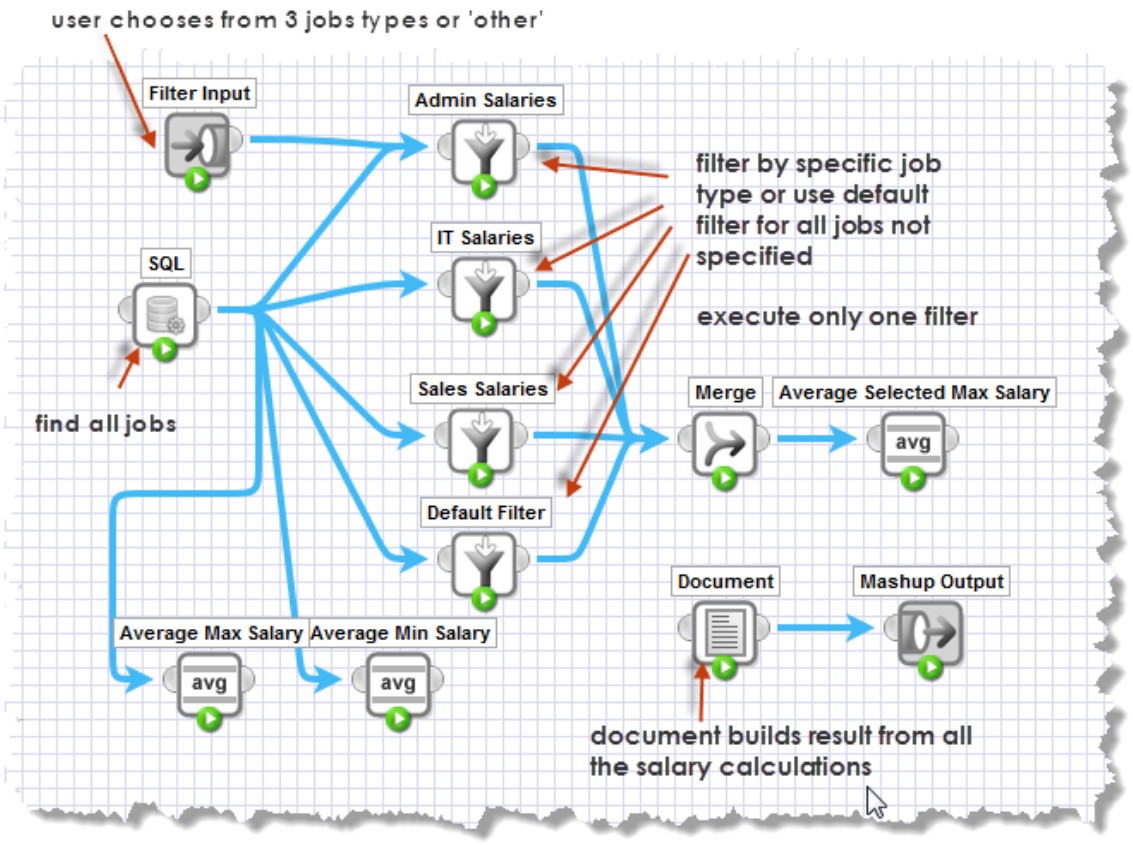


The "else clause" also consists of one Transformer block that executes if there are no issues from construction and builds somewhat different results.



Only one of the Transformer blocks will execute when the mashup is run, so the Merge block simply makes sure that the results that are actually generated become the mashup results.

The next example has multiple conditions based on a user selecting either one of three specific job types or selecting no job type.



There are three "if clauses" which consist of a Filter block to filter the list of jobs to jobs of a type that matches the input parameter. The execution conditions for each Filter block check to see if the value of the input parameter matches the job type for this Filter block. If so, the filter executes.

The "else" clause is the final Filter block. In this case, the execute condition for this Filter block tests to make sure that the value of the input parameter is *not* one of the three job types covered by the other Filter blocks.

Merge makes sure that the results from whichever Filter block actually executes is used and this becomes the input to a calculation. The final output of the mashup is built in the Document block, mapping fields to the various calculations.

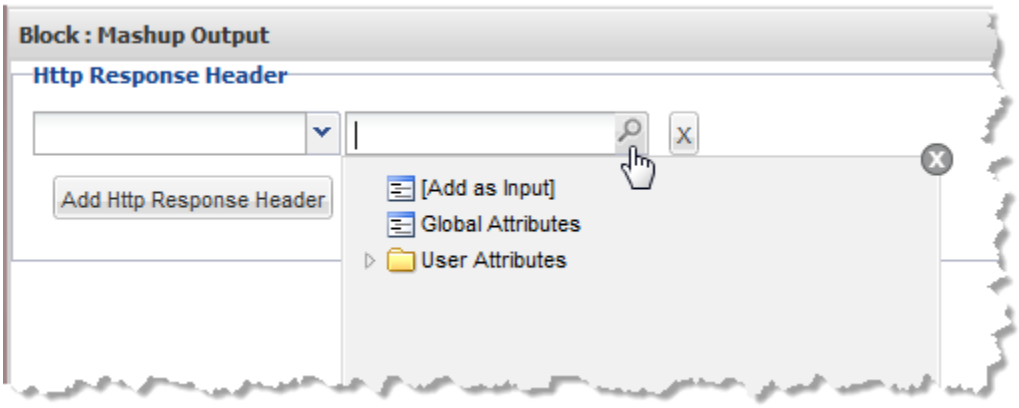
Add HTTP Headers to a Mashup

In some cases, you may need to add additional instructions to a mashup response using HTTP headers. These headers provide additional information about the response, such as the language of the response, instructions for caching and many others.

To add an HTTP response header for a mashup in Wires

1. Select the **Output** block on the canvas.

2. Click the Advanced Properties tab in the block properties pane.
3. Click **Add HTTP Response Header** to open fields to specify one HTTP header.



You can add any number of HTTP headers to the mashup.

4. Choose the name of the HTTP header from the left field.
5. Enter the value you want to use for this HTTP header or click to set the value for the header dynamically. See "Select Fields or Paths for Block Properties with the Path Selector" on page 372 for more information.

Advanced Properties: Controlling Blocks

Using the Advanced Properties tab, you can control the following aspects of how blocks within a mashup run:

- Add If/Else Conditions: using the Execute Condition section to set one or more conditions that must be met for a block to be run when the mashup is used. See for instructions and examples.

  Note: When conditions are not met during use of the mashup, the block does not run. This stops the flow of results from that block and can cause the mashup to have no results or cause other unexpected behavior.

- Define Error Handling: to determine how the mashup should react if there are errors or no responses from mashables or mashups that are used in this mashup.
- Add HTTP Headers to a Mashup: using the HTTP Headers section to define specific HTTP headers that should be included in the response for this mashup.
- Handle Empty Fields for WSDL Web Services: to determine how empty, optional fields should be handled in requests to WSDL Web Services.
- Use a Security Profile with DirectInvoke: to provide additional authentication information or other security configuration required for secured connections with some web sites or web services with the DirectInvoke action.

- Customizing the Selection XPath for Merge: to handle performance problems.

Define Error Handling

You can control how errors should be handled for mashables or mashups that are used in your mashup using the **Error Handling and Timeout** section of the Advanced Properties tab. This section only appears for mashable or mashup blocks.

Timeout	Set the number of seconds that MashZone NextGen should wait for a response from a mashable or mashup before continuing with processing the flow of this mashup.
On Error	Choose the action MashZone NextGen should take if the response from the mashable or mashup is an error. Be default, this is `Abort` which stops any further processing for this mashup. `Continue` allows the mashup to continue processing.

Handle Empty Fields for WSDL Web Services

Use the **Operation Request** section in Advanced Properties to determine how optional, empty fields are handled in requests for WSDL Web Services.

Empty fields	Change this property to either include or omits optional, empty fields in the requests sent to WSDL Web Services when they are invoked.

Use a Security Profile with DirectInvoke

Some web sites or web services require that the connection be secured or that you provide additional authentication information in the request when you request information from them or request other actions. MashZone NextGen uses *security profiles* to fill these requirements from information sources.

Note: A security profile is *not* required to provide secured connections for web sites or web services that use *one-way SSL* with the HTTPS protocol (in their URL). MashZone NextGen administrators must configure MashZone NextGen for one-way SSL, but no additional information is required. for more information.

Use the **Security Profile** section in Advanced Properties to supply this information for the DirectInvoke action.

Name	Enter the name of the security profile you need to use.

	Important: Currently DirectInvoke *only* supports the `CAS2` security profile.
Add Param	Click this button to add a parameter to send with this security profile. For CAS2, you only need to add a parameter if the verification URL for CAS for this web site or web service is *not* the URL (the endpoint) for the web site or web service. If they are different, add a parameter named `casServiceUrl` and enter the URL that the CAS Server should use to verify tokens for this web site or web service.

Customizing the Selection XPath for Merge

When you use the **Merge** action block and select the items to be merged, the XPath expression that defines these items can occassionally create performance problems for the mashup, especially if the items to merge are recursive within the results to be merged.

MashZone NextGen developers familiar with XPath can use the **Merge Path** property in Advanced Properties for the **Merge** block to directly edit this expression if needed.

Connect Mashup Blocks

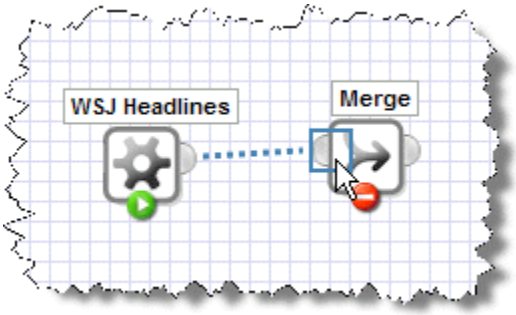
You define how the mashup works by the connections you draw between mashup blocks. A connection defines how the results from one block flow to other blocks and eventually become the final result of the mashup.

You draw connections starting from the *output port* of the block that is sending information to the *input port* of the block that is receiving information. It is also helpful if you preview the block that is sending information before you draw the connection.

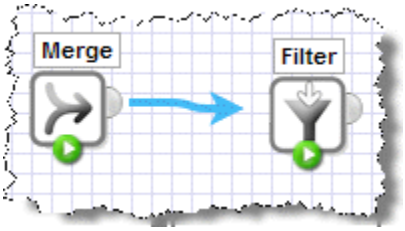
1. Hover the mouse over the output port of a block to start the connection.

   You should see an outline around the port.

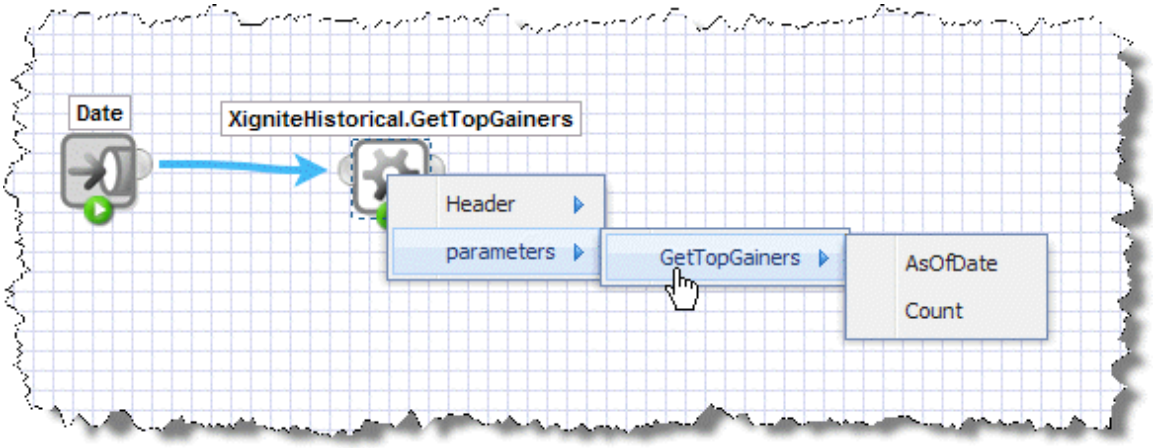
2. Hold the mouse button down and drag a line to the input port of the receiving block until it also displays a connection highlight.



3. Release the mouse button to complete the connection.



If the block receiving input from the connection has several input or header fields, you must choose which input or header field to connect to.



4. If desired, click **Preview** on the receiving block to see the results of this connection.

Advanced Updates with the Output Block

The **Output** block represents the final result for a mashup. You can use properties and advanced properties as short cuts or to perform more advanced configuration including:

- Quickly Set All Inputs for the Mashup
- Change the Mashup Result Character Encoding
- Add HTTP Headers to a Mashup

Quickly Set All Inputs for the Mashup

The block properties for the **Output** block is simply a list of all input fields for this mashup. You can use this to quickly set all the input parameters and apply them.

Change the Mashup Result Character Encoding

Character encodings define the set of characters used in the mashup result. By default, mashup results use a character encoding known as UTF-8. In some cases, you may need to change the character encoding so that characters in languages other than English are handled properly.

You can change the mashup character encoding in the Advanced Properties tab for the **Output** block.

Output Encoding	Select the character encoding to use for data in the result of this mashup. The list of valid character encodings is based on the version of Java used with the MashZone NextGen Server. See "Commonly Supported Output Character Encodings" on page 461 for descriptions of some of the most common encodings.
Mashable	If the data type is a custom, named type, enter the name of the WSDL web service that defines this named type.

Symbols to Escape in Block Properties

You must use substitute characters - an escaped version - for the following characters when you enter values in any block property:

Characters to Escape	Substitute Characters
<	<
>	>
&	&

Although it is not visible in Wires, mashups are saved as XML data. These three characters must be escaped inside XML data as they are also used as delimiters in XML.

Date and Time Formats, Math and Sorting in Wires

The format of characters used to represent dates or times can be quite varied which can affect their use in mashups in Wires. See "Date/Times As Inputs to Mashables or Mashups" on page 386, "Converting Other Formats to Datetime" on page 386, "Date/Times For Date Math" on page 386 or "Date/Times For Sorting" on page 386 for more information.

Date/Times As Inputs to Mashables or Mashups

The formats that you can use for date or time inputs to mashables depend on what individual mashables support. In most cases, you should consult mashable source documentation or ask your MashZone NextGen administrator for information on valid date and time formats.

For inputs to mashups, MashZone NextGen uses the *datetime* format: `YYYY-MM-DDThh:mm:ss.szzzzzz`. This is the XML dateTime datatype which is based on the ISO 8601 standard. If no time information is defined in the input, MashZone NextGen treats this as just a date. No default time is assigned.

Converting Other Formats to Datetime

In many cases, you can convert dates or times in other formats to the datetime format using the To Datetime function available in the **DataDecorator**, **Mapper** and **Transformer** blocks. The list of acceptable input formats is defined in the **Select input date format** property for this function.

Date/Times For Date Math

You can also calculate new dates or times by adding or subtracting a period of time from a date or time field using the Add to Date or Subtract from Date functions in the **DataDecorator**, **Mapper** and **Transformer** blocks. The base date to be updated must be a valid datetime. See "Converting Other Formats to Datetime" on page 386 for suggestions.

Date/Times For Sorting

You can sort block results based on dates or times in a variety of common formats. In general, however, the dates or times must include well known delimiters, such as / or - in dates, between the various components of the date or time. See "Date/Time Format Patterns" on page 353 for more information.

Manage Mashups in Wires

You use the Block Menus in Wires to find mashables, mashups or action blocks. You can also edit mashups from this menu. For instructions, see:

- "Quickly Find Mashables, Mashups or Other Blocks" on page 387
- "Edit a Mashup" on page 387
-
- "Edit Mashup XPath Expressions in Advanced Mode" on page 391
- "Save a Mashup" on page 388
- "Turn Mashups On or Off" on page 389
- "View Mashup or Mashable Details" on page 389
- "View Custom Block EMML Code" on page 390
- "Open the Mashup Artifact Page to Create Apps, Add Views or Take Snapshots" on page 390

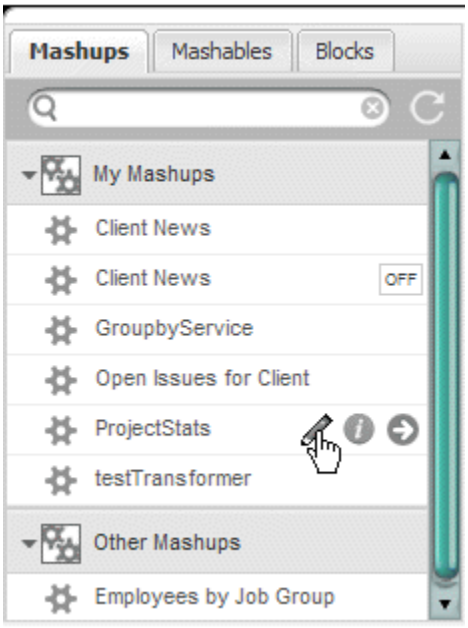
Quickly Find Mashables, Mashups or Other Blocks

Several tips to quickly find mashables, mashups or other blocks:

- To find mashups or mashables that you created, open the **My Stuff** Block menu and expand **My Mashables**.
- To work with your favorite mashups or mashables, open the **My Stuff** Block menu and expand **My Favorites**.
- Use **Search** to search all mashables or mashups by name, description or tag.
- For action and other blocks, open the **Blocks** menu and expand a category.
- To shorten the **My Stuff** or **Blocks** menu, enter part of a name, description or tag as a **filter**.
- Use to clear search or the filter.
- Use to refresh the **My Stuff** menu.

Edit a Mashup

1. Select your mashup in the Block Menus and click **Edit**.



If you have permission to edit this mashup, it opens in the Design Canvas. If the mashup is turned on, you must click ON to turn it off before you can edit the mashup.

> **Note:** If the mashup you selected was not created in Wires, you will see a message that Wires cannot open the mashup.

2. Change the blocks and connections as needed.
3. Click **Save**. If the mashup is complete (fully configured and connected), Wires automatically turns the mashup on.

Save a Mashup

When you save a mashup, Wires also automatically sets the status to ON unless either one or more blocks are not fully configured or the results from the final block are not connected to the Output block.

Mashups with an OFF status are available for further editing in Wires, but cannot be used in other mashups, apps or workspaces. Only you or a MashZone NextGen administrator can edit, turn on or delete mashups that are turned off.

> **Note:** Mashups that you have saved but not yet published display an *Inactive* flag in the Block Menus.

1. Click **Save**.
2. Enter a **Name** for the mashup.

MashZone NextGen uses the mashup name to assign a unique identifier to the mashup. Mashup names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

3. Enter or select optional information including:
   - A description of the purpose of this mashup.
   - The name of the information provider to assign to this mashup.
   - Tags for this mashup.
   - A category for this mashup.
4. Click **OK**.

Once your mashup is complete, you also need to grant permissions to other users to use it and add views to the mashup to use in apps, snapshots or workspaces.

Turn Mashups On or Off

Mashups in an off status can be edited, but cannot be used in other mashups or apps by other users. Turning a mashup on, also called publishing the mashup, makes it available for use.

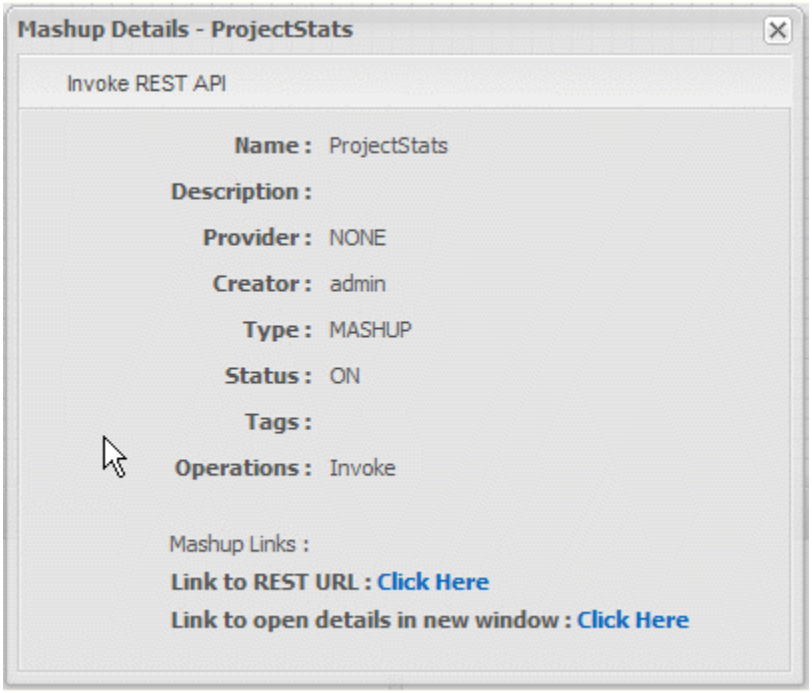
Use the

and

buttons to control this state.

View Mashup or Mashable Details

You can see more information for any mashable or mashup listed in the Block Menus. Hover over the mashable or mashup and click **Information**.



Click **Open Details in New Window** to view more information, add views, take snapshots, create apps or perform other tasks.

View Custom Block EMML Code

Some custom action blocks are created using EMML macros. You can view the EMML code for these blocks by clicking **View macro EMML code** in the Blocks Menu.

Open the Mashup Artifact Page to Create Apps, Add Views or Take Snapshots

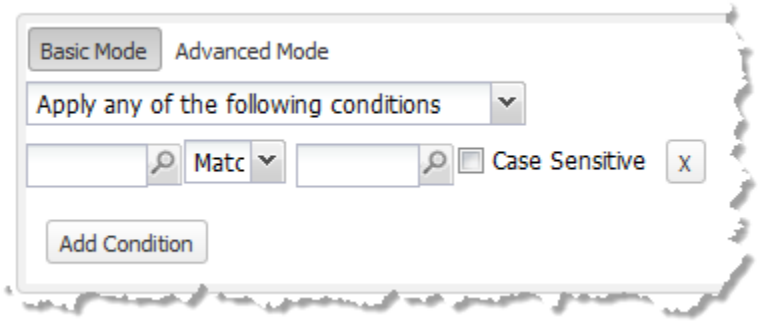
Once you have turned a mashup on, you work with the mashup in its artifact page to add views, create apps or take snapshots. You also grant permissions to other users to run your mashup from its artifact age.

You can open a mashup artifact page from Wires using the **Open Mashup** toolbar button or click **View Information** in the Blocks Menu to find a link in the information window.

See "Grant Permission to Run Mashables, Mashups and Apps" on page 102, "Add Views to Mashables and Mashups" on page 901, "Create a Basic App" on page 1093, "Take, View or Delete Snapshots" on page 183 and "Schedule Snapshots" on page 185 for more information on these tasks.

Edit Mashup XPath Expressions in Advanced Mode

The **Advanced Mode** and **Basic Mode** buttons appear in block properties for built-in MashZone NextGen action blocks that use conditions, such as **Filter** and **Join**. They also appear in Advance Properties for **Execute Conditions** for all mashables and mashups, most of the built-in MashZone NextGen action blocks and any custom action blocks that enable conditional execution.

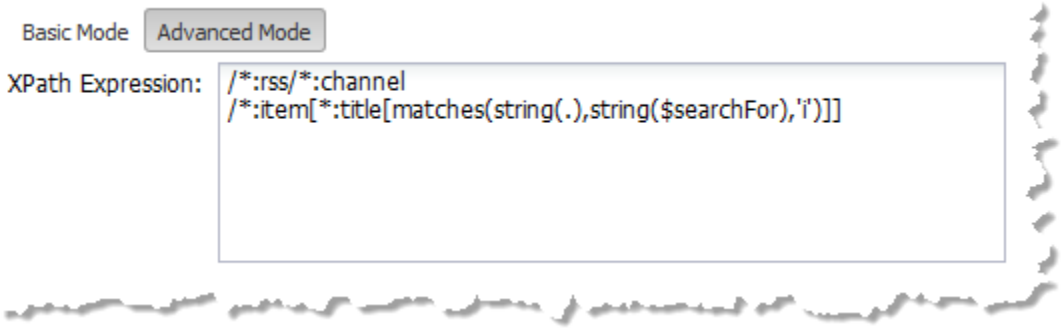


Basic mode is simply the existing properties you see for a block or execute condition. Advanced mode, however, allows MashZone NextGen developers to directly edit the XPath expressions for a condition.

Important: The advanced mode *should be used with care* by developers who are knowledgeable in XPath syntax. Editing XPath expressions directly can cause errors in the mashup.

Moreover, once a condition is in advanced mode in Wires, returning to basic mode completely removes the XPath expression and the condition.

To edit XPath expressions for conditions, simply click **Advanced Mode** in the Block Properties pane or the Advanced Properties tab.



Edit the expression as needed, following XPath syntax. For more information on XPath, see the XPath 2.0, XPath 2.0 functions and the XPath 2.0 Data Model specifications.

Customizing Wires

MashZone NextGen developers can create custom blocks that are specific to your needs for use in Wires. MashZone NextGen administrators can also add XML schemas for use with the Mapper block to easily map results to well-known formats. For more information, see:

- "Adding Custom Blocks to MashZone NextGen Wires Using Macros" on page 392

Adding Custom Blocks to MashZone NextGen Wires Using Macros

You can extend the mashup capabilities of MashZone NextGen Wires by adding custom action blocks with logic to meet your specific needs. Custom action blocks are defined as macros - custom EMML statements - with additional metadata that allows Wires to load and work with the macro.

Note: This topic includes two examples to take you through:

- A simple macro meant for use with specific, well-known results
- A generic macro that can handle a wider variety of results.

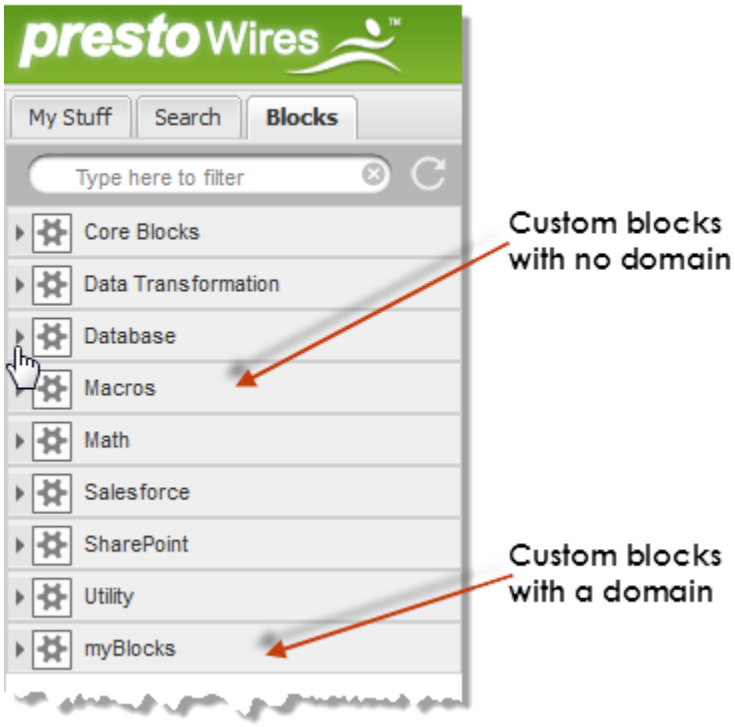
To create a custom action block in Wires, you must create the macro with appropriate code to enable its use in Wires and then save and register it with the MashZone NextGen Server. See "Creating and Registering Macros" on page 643 for the basic steps involved in creating and registering macros. See "<macro>" on page 638 for basic instructions on defining macro logic.

When a macro will be used as a custom Wires block, there are additional requirements and optional designs to consider:

- "Use Domains to Organize Custom Blocks" on page 392
- "Define Inputs and Output for 'Wiring' the Macro" on page 393
- Handle common use cases for blocks, including:
  - "Loop Through a Well-Known Input" on page 394
  - "Make Macros Generic for Custom Blocks" on page 396
  - "Use or Construct XPath Expressions for Generic Macros" on page 397
- Configure the appearance and behavior of the custom Wires block for this macro. See "Configure Properties for Custom Blocks" on page 403 for details.

Use Domains to Organize Custom Blocks

All custom blocks appear on the Blocks tab of the Block Menus in Wires. This menu has a set of built-in categories to help organize blocks and make it easier for users to find what they are interested in.



If you register macros used as custom blocks without a *domain*, the blocks are added to a category named `Macros`. This single category can quickly be overwhelmed with custom blocks.

Instead, it is a best practice to organize custom blocks into additional categories to help identify their purpose or scope. You define the category for a custom block by assigning a domain to the macro. In addition to organizing custom blocks, macro domains also ensure that macro names are unique.

For macros defined in a macro library, you assign the domain to the entire library of macros:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
...
</macros>
```

For macros you define individually in the Mashup Editor, you assign the domain name when you save the macro.

Define Inputs and Output for 'Wiring' the Macro

The macro will end up as a block in a Wires graph that is connected to other blocks. To enable 'wiring', the macro must have at least one <input> statement that can be wired to other blocks. The macro must also have an <output> statement to allow the results of the block to be wired to another block in the mashup. For example:

```
<macros xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/ ../src/schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    domain = "myCustomMath">
 <macro name="GetSessionPercentages">
    <!-- Spreadsheet with Session data as input -->
    <input name="spreadsheet" type="document" />
    <output name="result" type="document" />
    ....
  </macro>
...
</macros>>
```

In many cases, some of the <input> statements are a document type to accept complex, structured results from other blocks. The <output> statement also typically is a document type. However, this is not required. Both <input> and <output> statement types can be string or any other type that meets your needs.

This next example works with a 'well-known' input, in this case a spreadsheet mashable. You can also design macros to work with generic inputs (see "Make Macros Generic for Custom Blocks" on page 396).

Loop Through a Well-Known Input

One very common use case for custom blocks is to take results from a mashup or mashable information source, loop through each item in the results and add or transform data in the results.

When the structure of the input is well known, common steps that a looping macro may need to complete are:

1. Define <output> with a root node that will wrap the items added by the macro during looping. See "Creating a Root Node for the Macro Result" on page 394.
2. Begin looping through the input. See "Looping With <foreach> and a Well-Known Input Structure" on page 395.
3. Generate the appropriate output for each loop. See "Using <appendresult> to Transform Input and Build the Output" on page 396.

Creating a Root Node for the Macro Result

You must add an <output> statement for the macro to allow the custom action in Wires to be wired to other blocks or to the output of the mashup itself. When a macro loops through the input block results, you can also create the root node for the macro output within the <output> statement itself. Simply add an empty element within <output>.

This example defines a root node, <statistics>, for the macro output:

```
<macros xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/ ../src/schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    domain = "myCustomMath">
  <macro name="GetSessionPercentages">
    <!-- Spreadsheet with Session data is input -->
    <input name="spreadsheet" type="document" />
```

```
   <!-- create wrapper for results that macro will append to -->
    <output name="result" type="document">
      <statistics />
    </output>
    ...
  </macro>
...
</macros>>
```

Looping With <foreach> and a Well-Known Input Structure

You use <foreach> to loop through a set of repeating items, in this case the input mashable results.

With <foreach> you define a variable to hold each item and use an XPath expression to define the items to loop through. The input for this example comes from an spreadsheet mashable:

```
<SessionStatsData>
  <item>
    <Categories>FormA</Categories>
    <Session1>23</Session1><Session2>30</Session2><Session3>35</Session3>
    <Session4>11</Session4><Session5>23</Session5><Session6>15</Session6>
  </item>
  <item>
    <Categories>FormC</Categories>
    <Session1>15</Session1><Session2>10</Session2><Session3>13</Session3>
    <Session4>15</Session4><Session5>5</Session5><Session6>11</Session6>
  </item>
  <item>
    <Categories>FormD</Categories>
    <Session1>37</Session1><Session2>78</Session2><Session3>51</Session3>
    <Session4>66</Session4><Session5>77</Session5><Session6>25</Session6>
  </item>
  <item>
    <Categories>FormL</Categories>
    <Session1>8</Session1><Session2>3</Session2><Session3>6</Session3>
    <Session4>4</Session4><Session5>5</Session5><Session6>6</Session6>
  </item>
  <item>
    <Categories>FormR</Categories>
    <Session1>19</Session1><Session2>22</Session2><Session3>7</Session3>
    <Session4>15</Session4><Session5>10</Session5><Session6>13</Session6>
  </item>
</SessionStatsData>
```

The <foreach> loop would look something like this:

```
<macros xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/ ../src/schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    domain = "myCustomMath">
  <macro name="GetSessionPercentages">
    <input name="spreadsheet" type="document" />
    <output name="result" type="document">
      <statistics />
    </output>
    <!-- loop through items in spreadsheet -->
    <foreach variable="$category" items="$spreadsheet//item">
      ...
    </foreach>
  </macro>
```

```
...
</macros>>
```

Within the loop, the `category` variable holds each item during an iteration. The XPath that defines what node to use as an item is specifically defined based on the well known structure of the input.

Using <appendresult> to Transform Input and Build the Output

Within the loop, you add to the output using <appendresult>. You define the structure of the output as literal XML and use dynamic mashup expressions to perform calculations, transform data and fill the output. See "Dynamic Mashup Expressions" on page 604 for basic instructions.

This example adds a <category> element for each item in the loop and calculates percentages for the session statistics. The content for <appendresult> is again specifically defined from the known structure of the input.

```
<macros xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/ ../src/schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    domain = "myCustomMath">
  <macro name="GetSessionPercentages">
    <input name="spreadsheet" type="document" />
    <output name="result" type="document">
      <statistics />
    </output>
    <!-- loop through items in spreadsheet -->
    <foreach variable="$category" items="$spreadsheet//item">
      <!-- add a category and calculate percentages for each item -->
      <appendresult outputvariable="$result">
        <category>
          <name>{$category/Categories/string()}</name>
          <session1PC>{$category//Session1/number() div 100}</session1PC>
          <session2PC>{$category//Session2/number() div 100}</session2PC>
          <session3PC>{$category//Session3/number() div 100}</session3PC>
          <session4PC>{$category//Session4/number() div 100}</session4PC>
          <session5PC>{$category//Session5/number() div 100}</session5PC>
          <session6PC>{$category//Session6/number() div 100}</session6PC>
        </category>
      </appendresult>
    </foreach>
 </macro>
...
</macros>>
```

Make Macros Generic for Custom Blocks

You can design macros for custom blocks to work with the results of a wider range of mashables and mashups - to be generic. Defining the logic of a generic macro is the same as any macro. There are, however, two issues that generic macros typically must take into account:

- Do you need to accommodate block results that have XML namespaces as inputs?

  Since users may connect many different types of results as inputs, the XPath expressions within the macro should be prepared to handle results that use

namespaces. You can use wildcards as a namespace in the XPath expressions, such as `$someVar/*:root/*:child`.

- How does the macro identify the nodes it must work with for the different block results that are used in different mashups?

  Since you do not know the structure of the input when you design the macro, users must identify the nodes that the macro will work with from the block(s) that they connect as input to the custom block for the macro. So, in addition to having one or more input parameters to receive block results, the macro must have input parameters that identify the *nodes of interest* to the macro.

  These 'nodes of interest' are identified by XPath expressions for the specific input block results in a given mashup where the macro is used. This results in requirements for the macro and for its custom block in Wires:

  - Users must be able to easily supply these XPath expressions when they use the custom block in Wires.

    The solution to this issue is to force users to select the appropriate nodes in the **Path Selection** list in Wires and then assign the XPath expressions for the selected nodes to the block properties for the appropriate macro input parameters.

    You use metadata configuration in the macro to define this behavior for the custom block. See "Get the Chosen Path as Text" on page 426 for information on the appropriate metadata.

    Note: In previous releases, you could define which input parameters should contain XPath expressions using a `<presto:presto-meta>` statement. Effective with MashZone NextGen 3.2, this syntax is *deprecated*.

  - The macro must use these XPath expressions within its logic statements to work on the specific results for that mashup.

    This involves using the XPath expression or constructing XPath expressions dynamically and using them. See "Use or Construct XPath Expressions for Generic Macros" on page 397 for instructions and examples.

Use or Construct XPath Expressions for Generic Macros

You use the <type> and <xpath> metadata elements within <presto:macro-meta> to have Wires force users to select nodes from the **Path Selection** list for an input parameter and assign the XPath expression for the selected node to a macro input parameter. The next example shows the basics of this metadata:

```
<macro name="helloWorld"
     xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
  <presto:macro-meta>
    <block usages="Wires"> ... </block>
    <parameters>
      <parameter name="aPath">
        <type datatype="path">
          <xpath limitTo="$source" usage="leaf"/>
        </type>
      </parameter>
```

```
    </parameters>
  </presto:macro-meta>
  <output name="macroResult" type="document"/>
  <input name="source" type="document"/>
  <input name="aPath" type="string"/>
  ...
  <sort />
</macro>
```

For more information on this metadata, see "Get the Chosen Path as Text" on page 426.

To use the XPath expression from the input parameter, you generally need to include the name of the input parameter that contains the document-type results where this path is taken from. You may also need to provide additional information to get the complete XPath expression you need to use in your macro.

For examples, see:

- "Modifying XPath Expressions to Handle Generic Loops" on page 398 and "Custom Block Properties and Metadata for Generic Loop Example" on page 399
- "Modifying XPath Expressions to Enable Generic Extraction" on page 402

Modifying XPath Expressions to Handle Generic Loops

In this generic macro example, the macro loops through a repeating set of leaf nodes to convert dates from an invalid format into valid dates that can be used in a mashup for sorting or in date calculations. This example is meant to be used with information sources that have dates as a string of numbers with no delimiters, such as 20100904.

Users identify the set of repeating leaf nodes with dates that should be converted in the `pathToDate` input parameter. This path is then used in a <foreach> statement in the macro to perform the conversion.

The `items` attribute of <foreach> expects an XPath expression in the form `$`*variable-name*/*path-within-the-variable* . The `pathToDate` input parameter, however, contains just /*path-within-the-variable* .

To handle this, the XPath expression in `items` uses a dynamic mashup expression (see "Dynamic Mashup Expressions" on page 604) to construct the full path needed using the path passed in `pathToDate`:

```
<macro name="ToDate"
     xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
  <presto:macro-meta>
    ...
  </presto:macro-meta>
  <output name="macroResult" type="document"/>
  <input name="source" type="document"/>
  <input name="pathToDate" type="string"/>
  <input name="dateFormat" type="string"/>
  <foreach items="$source{$pathToDate}" variable="aDate">
    <variable name="year" type="string"/>
    <variable name="month" types="string"/>
    <variable name="day" type="string"/>
    <if condition="$dateFormat='1'">
      <assign fromexpr="substring($aDate/text(),1,4)" outputvariable="$year"/>
      <assign fromexpr="substring($aDate/text(),5,2)"
```

```
        outputvariable="$month"/>
      <assign fromexpr="substring($aDate/text(),7,2)" outputvariable="$day"/>
    <elseif condition="$dateFormat='2'">
      <assign fromexpr="substring($aDate/text(),1,2)"
        outputvariable="$month"/>
      <assign fromexpr="substring($aDate/text(),3,2)"
        outputvariable="$day"/>
      <assign fromexpr="substring($aDate/text(),5,4)" outputvariable="$year"/>
    </elseif>
    <else>
      <assign fromexpr="substring($aDate/text(),1,2)" outputvariable="$day"/>
      <assign fromexpr="substring($aDate/text(),3,2)"
        outputvariable="$month"/>
      <assign fromexpr="substring($aDate/text(),5,4)" outputvariable="$year"/>
    </else>
    </if>
    <assign fromexpr="concat($year,'-',$month,'-',$day)"
      toexpr="$aDate"/>
  </foreach>
  <assign fromvariable="$source" outputvariable="$macroResult"/>
</macro>
```

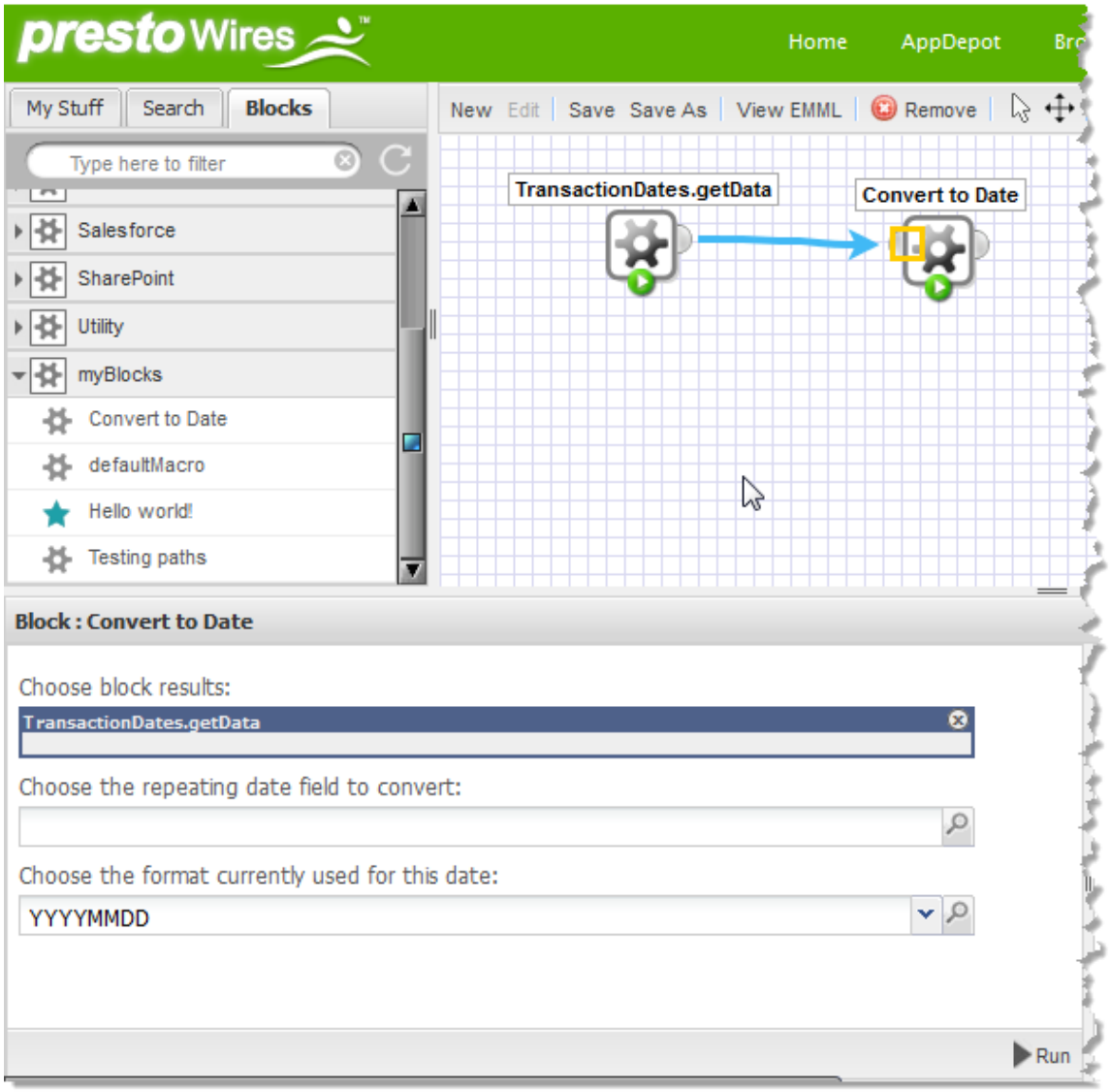
The loop then deconstructs each date, based on the format identified in the `dateFormat` input parameter, reconstructs a date in the ISO 8601 format that EMML uses and updates the date field with the reconstructed date.

Custom Block Properties and Metadata for Generic Loop Example

The ToDate example shown in the generic loop macro has three input parameters:

- `source`: to receive the document with dates to convert.
- `pathToDate`: to pass the XPath expression for the nodes that users choose that identifies the repeating dates nodes to convert.
- `dateFormat`: a choice of date formats to identify the format of the incoming dates.

The custom block properties for this macro looks like this:

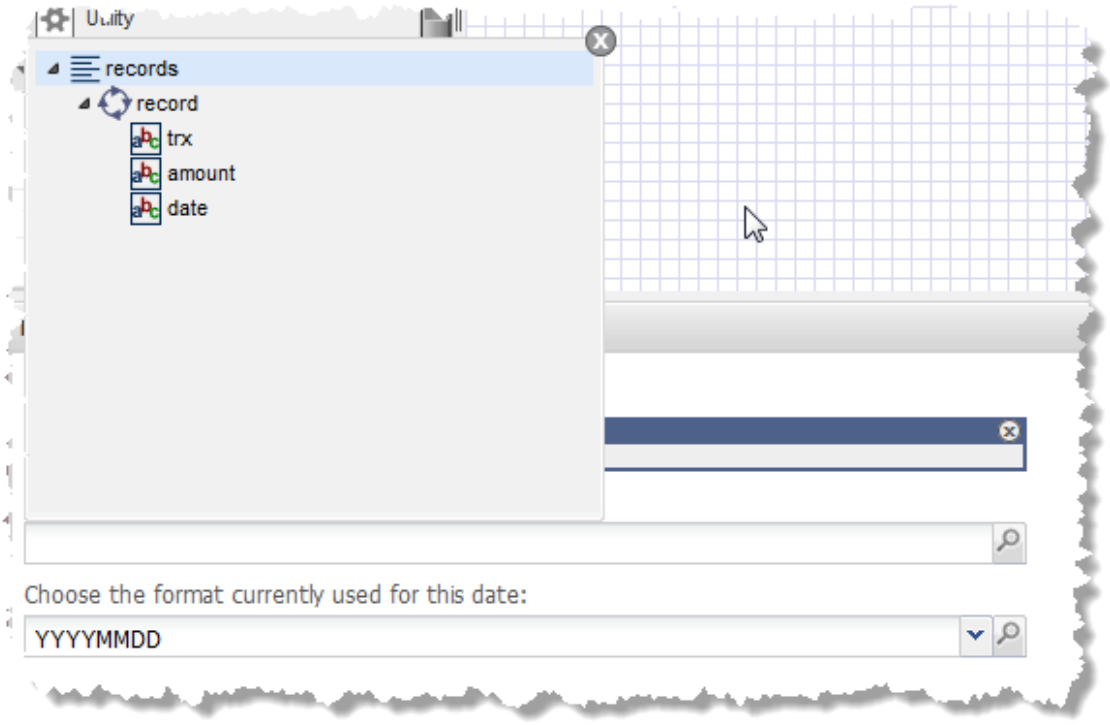


To ensure that users select fields with data for `pathToDate` and that what is passed is the XPath expression to the nodes they select, you supply metadata in <presto:macro-meta>, such as this:

```
<macro name="ToDate"
     xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
  <presto:macro-meta>
    <block usage="Wires">
      <label>Convert to Date</label>
    </block>
    <parameters>
      <parameter name="source">
        <label>Choose block results</label>
      </parameter>
      <parameter name="pathToDate">
        <label>Choose the repeating date fields to convert</label>
        <type datatype="path">
```

```
          <xpath limitTo="$source" usage="leaf"/>
        </type>
      </parameter>
      <parameter name="dateFormat">
        <label>Choose the format currently used for this date</label>
        <type datatype="enum">
          <list>
            <option label="YYYYMMDD">1</option>
            <option label="MMDDYYYY">2</option>
            <option label="DDMMYYYY">3</option>
          </list>
        </type>
      </parameter>
    </parameters>
  </presto:macro-meta>
  <output name="macroResult" type="document"/>
  <input name="source" type="document"/>
  <input name="pathToDate" type="string"/>
  <input name="dateFormat" type="string"/>
...
</macro>
```

In this example, <type datatype="path"> indicates that the `pathToDate` input parameter should receive the XPath expression rather than the value of the selected node. The <xpath> statement identifies the results that users must choose from in the **Path Selection** list for `pathToDate`:

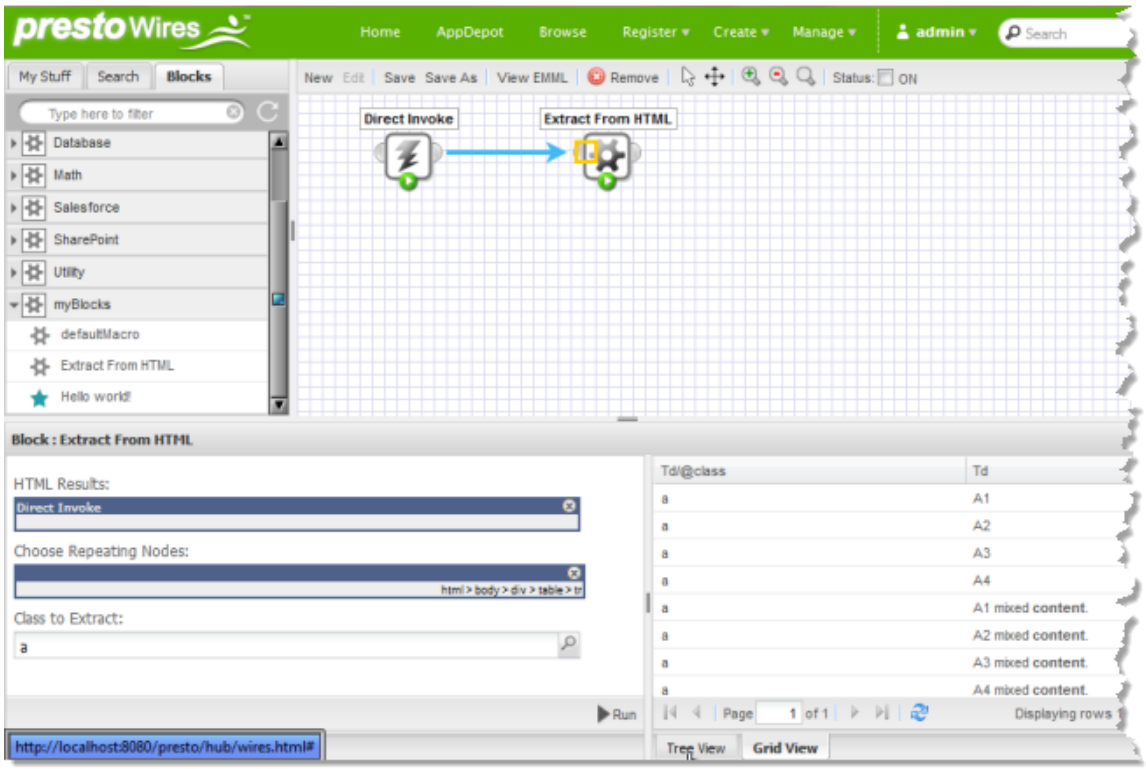


For more information and links on the metadata that you can use to configure block properties for custom blocks, see "Configure Properties for Custom Blocks" on page 403.

Modifying XPath Expressions to Enable Generic Extraction

The built-in MashZone NextGen Extract block allows users to extract data values or nodes primarily based on their position in results. In many cases, however, you need more sophisticated logic to extract nodes based on node names or attribute values.

This generic macro example allows users to extract nodes from HTML results based on the value of class names in the HTML. A common use for this is to allow users to use *web clipping* to retrieve information from web sites.



In this example, the Direct Invoke block retrieves an HTML page. The Extract From HTML custom block, based on this example macro, has three input parameters to identify the block results to extract nodes from, to identify a parent or ancestor that contains all the nodes that users want to extract and to identify the class name that should be used to extract nodes.

The code for this example, along with the metadata that configures the custom block properties in Wires, is:

```
<macro name="extractByClass"
     xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
 <presto:macro-meta>
    <block usage="Wires"><label>Extract From HTML</label></block>
    <parameters>
      <parameter name="htmlDoc">
        <label>HTML Results</label>
        <help>Connect a block with HTML content</help>
      </parameter>
```

```
      <parameter name="path">
        <label>Choose Repeating Nodes</label>
        <help>Choose the repeating ancestor for the nodes to extract</help>
        <type datatype="path">
          <xpath limitTo="$htmlDoc" usage="array"/>
        </type>
      </parameter>
      <parameter name="extractClass">
        <label>Class to Extract</label>
        <help>Enter the class that identifies the nodes to extract</help>
      </parameter>
    </parameters>
  </presto:macro-meta>
  <output name="macroResult" type="document"/>
  <input name="htmlDoc" type="document"/>
  <input name="path" type="string"/>
  <input name="extractClass" type="string"/>
  <variable name="temp" type="document">
    <extracted/>
  </variable>
  <foreach items="$htmlDoc{$path}//*[contains(@class,'{$extractClass}')]"
      variable="record">
    <appendresult outputvariable="$temp">
      <record>{$record}</record>
    </appendresult>
  </foreach>
  <assign fromvariable="$temp" outputvariable="$macroResult"/>
</macro>
```

The macro uses a dynamic mashup expression in the <foreach> loop to combine both the XPath expression from the `path` input parameter and the class name from `extractClass` to identify the HTML nodes to extract. For more information, see "Dynamic Mashup Expressions" on page 604.

The macro metadata for `path` ensures that the block property in Wires forces users to select a repeating node from the HTML results in `htmlDoc`. This property is assigned the XPath expression to the node that users select. For more information on this metadata configuration, see "Get the Chosen Path as Text" on page 426.

Configure Properties for Custom Blocks

By default, all macros generate custom blocks in Wires unless you *explicitly exclude* them. See "Example 106. Macro Metadata" on page 641 for instructions on how to exclude macros from Wires.

If no additional configuration is provided with a macro, Wires generates a default block for the Block Menus and the canvas. It also generates a default block properties form with fields for each input parameter in the macro. See "Default Custom Block With No Configuration" on page 404 for details on these defaults.

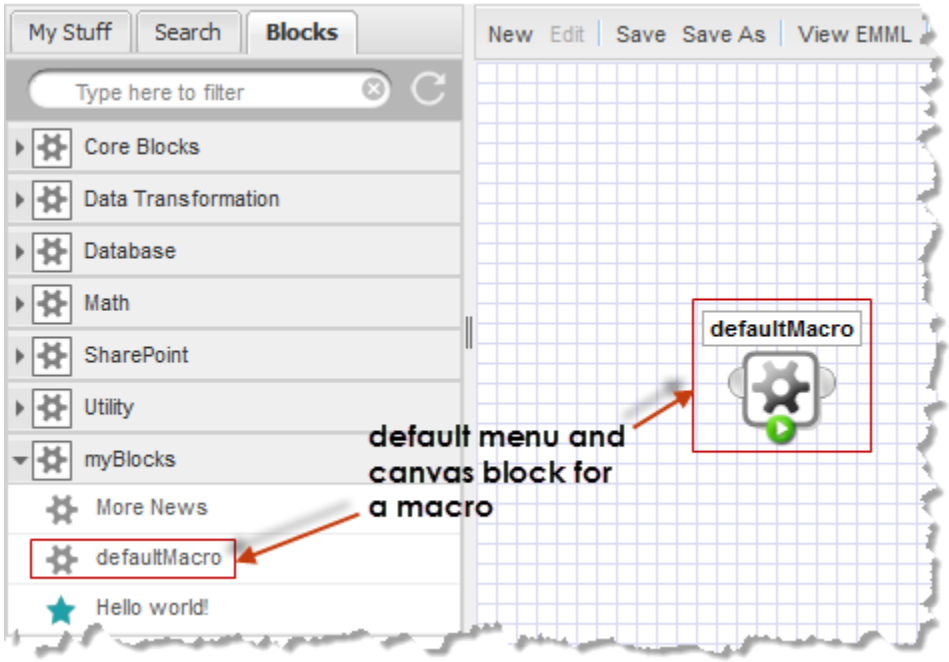
You can configure certain visual aspects or behavior for a custom block in Wires, using the "<presto:macro-meta>" on page 583 statement in the macro, including:

- "Update Labels and Icons for Custom Blocks in Wires" on page 406
- Change the order of properties. This is defined by the order that <input> statements are defined in the macro. Simply update the EMML to change the order of properties in the custom block.

- "Add Tooltips and Help in Custom Blocks" on page 409
- "Make Custom Block Properties Required" on page 411
- "Configure Field and Line Sizes for Custom Block Properties" on page 412
- "Configure Strings, Numbers, Dates or Boolean Properties in Custom Blocks" on page 415
- "Limit Custom Block User Entries to a List of Values" on page 429
- "Configure Dynamic Path Choices and Results for Custom Blocks" on page 423
- "Advanced Custom Block and Property Configuration" on page 435

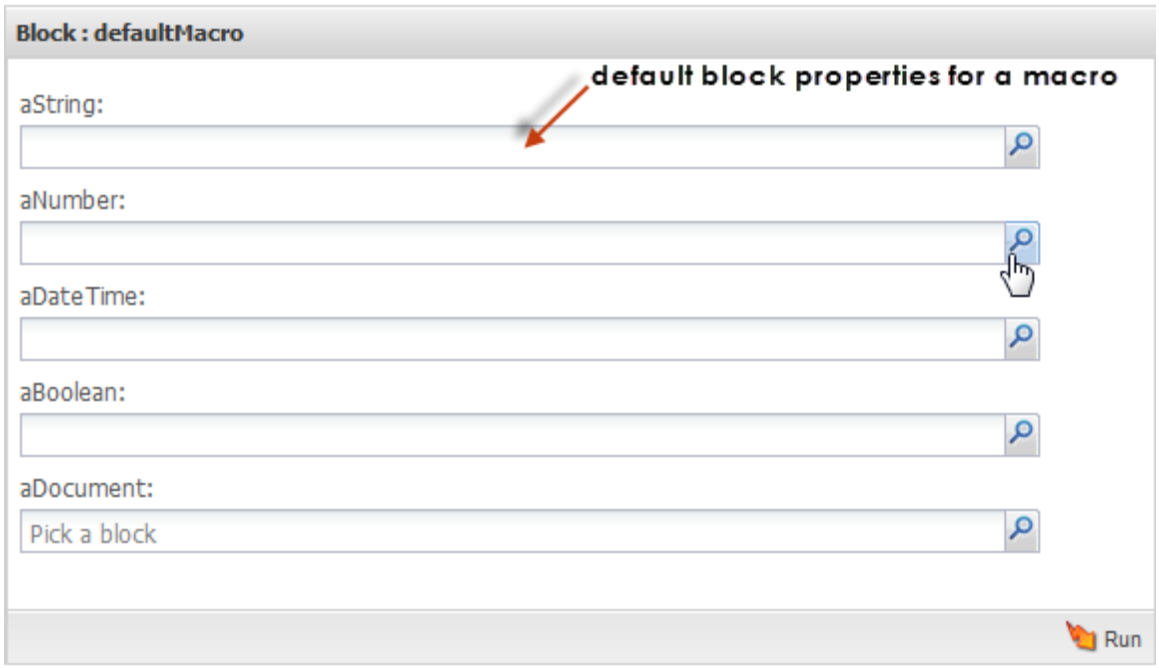
Default Custom Block With No Configuration

Macros that have no metadata for custom block configuration use the default Gear icon in Wires. The macro name becomes the label for block:



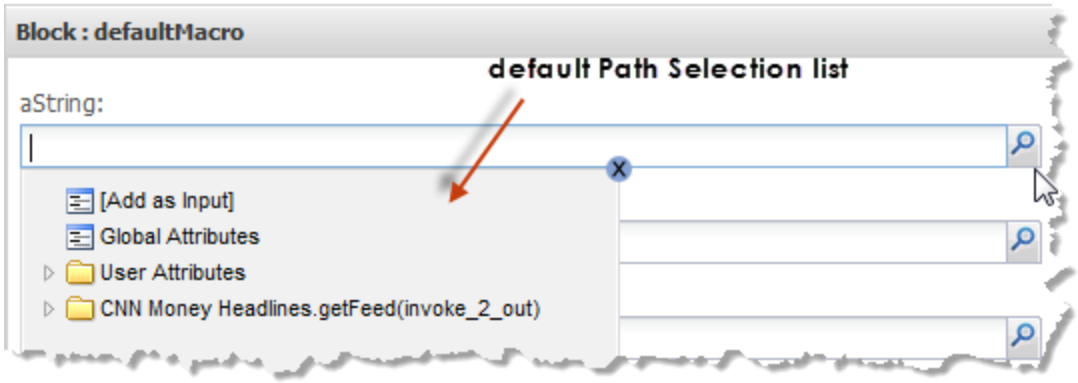
Macros appear in the Blocks tab in a category with their domain name or category named `Macros` if they have no domain.

The block properties form has a label and a property field for each input parameter in the macro, listed in the order in which <input> statements appear within the macro. Default property labels are the input parameter name:



Property fields are the same for all input parameters, regardless of their datatype: a combination of a single-line input field with the Path Selection button.

- For input parameters with any simple datatype, users can enter literal values in the block property.
- For input parameters with a document datatype, users can only select the results from an existing block in the mashup.
- For input parameters of any type, they can select dynamic values from the **Path Selection** list using input parameters for the mashup, any MashZone NextGen attribute or any node from the results of any block in the mashup:



There are no tooltips for the block or properties and no help available for the block. Limited validation is applied to user entries for properties, based on the datatype for the corresponding macro input parameter.

You can configure a variety of these aspects for custom blocks using the <presto:macro-meta> MashZone NextGen extension statement. The basic requirements for <presto:macro-meta> include a <block> child that defines the usage for this macro:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires"/>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

This example explicitly declares that the `helloWorld` macro is meant to produce a custom block in Wires in a menu category named `myBlocks`. See "Example 106. Macro Metadata" on page 641 for instructions on excluding macros from Wires.

Update Labels and Icons for Custom Blocks in Wires

You can change Block Labels for a custom block, Property Labels or Block Icons.

Block Labels

You can add configuration to change the default label for a custom Wires block within the macro that generates the block. Use the <presto:macro-meta> statement and add <block> and <label> wtihin <block>:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        <label>Hello world!</label>
      </block>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Tip: Custom labels can be any text, including spaces, that is valid in XML. Characters that are XML delimiters, such as &, must be escaped.

It is a good practice to keep labels for custom blocks fairly short to ensure that they fit in the menu and do not take an excessive amount of space on the canvas. Length is less of an issue for custom property labels.

Property Labels

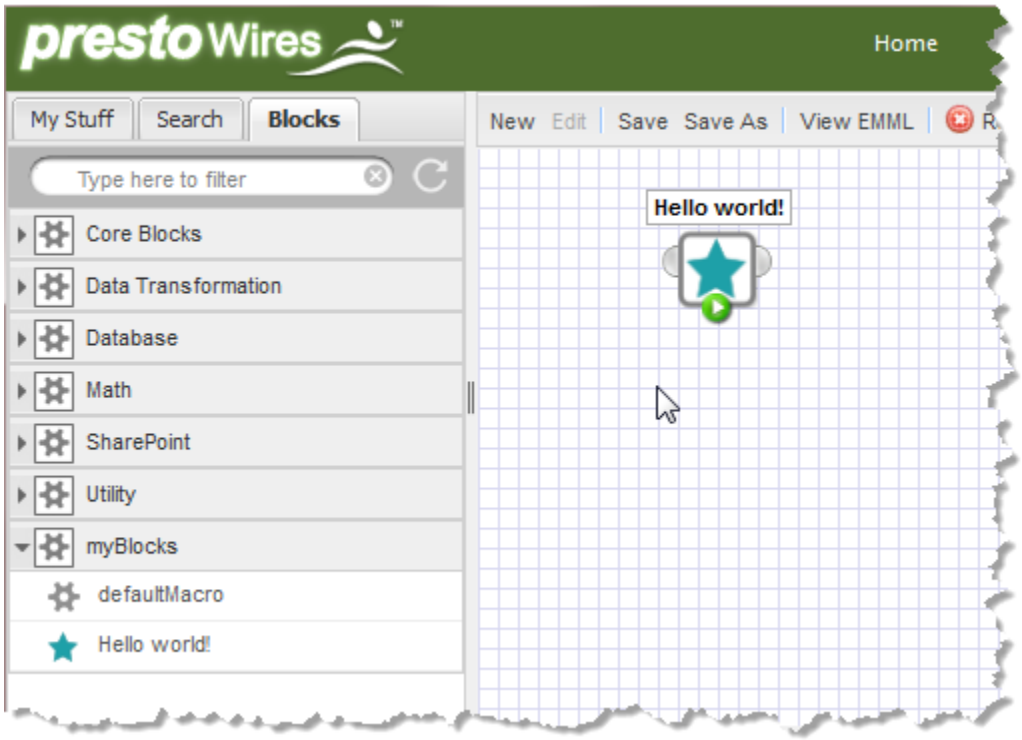
For property labels in custom blocks, you must add the macro metadata element <parameters> to <presto:macro-meta> and then add <parameter> with <label> to define the label for a specific property:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        <label>Hello world!</label>
      </block>
      <parameters>
       <parameter name="aString">
         <label>Your name</label>
       </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

The `name` attribute in <parameter> must match the name of the macro input parameter that this configuration should apply to.

Block Icons

You can also supply images to use as icons for a custom block using the macro metadata element <icon> in <block>.



If you supply a custom icon, you *must provide two images* of the correct size for the Wires Blocks Menu and for the canvas:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        <label>Hello world!</label>
        <icon usage="menu" url="http://myOrg.com/images/smallMacroIcon.png"/>
        <icon usage="canvas" url="http://myOrg.com/images/lgMacroIcon.png"/>
      </block>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Custom block icon images can be in any format that is web accessible. The size of the image for each icon is:

- *menu icon* = a maximum of 14 pixels square, with no border.
- *canvas icon* = exactly 40 pixels square, including a 2-3 pixel border. The built-in Wires blocks use an icon that is 38px square with a 2px drop shadow. The border is 2px wide in #818181 grey.

> **Tip:** Canvas icons should always have a border to ensure that input and output ports for the block render well and are clearly associated with that block.

As with all icons, the image should be graphic and easy to distinguish from the icons for other blocks.

The URL to icons can be absolute when the image is hosted in a web server or application server that is accessible to the MashZone NextGen Server.

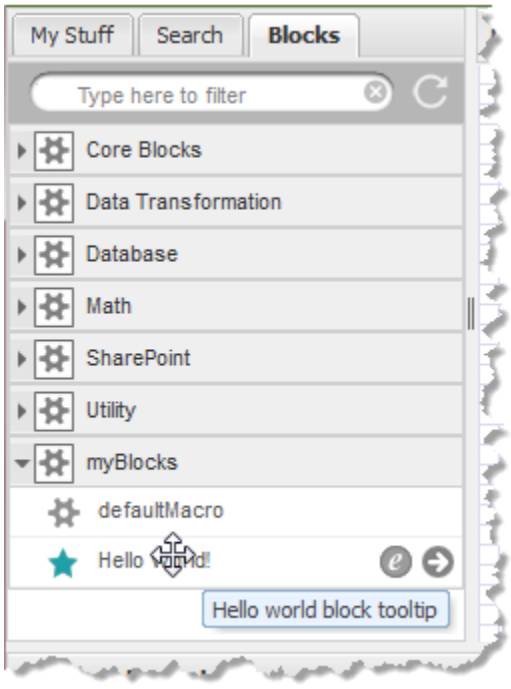
You can use relative URLs for icons if a MashZone NextGen administrator uploads the images. Use a URL in the form `/presto/files/`*path/filename* where *path/filename* is the name assigned to the image in MashZone NextGen.

Add Tooltips and Help in Custom Blocks

You can set Tooltips for the block or for individual properties. You can also Add a Help Button and Help Topic for the entire block.

Tooltips

Use the macro metadata element <help> in <block> or <parameter> to add tooltips for a custom Wires block or for individual properties in a custom block.



Tooltip text should be fairly short. It must be valid text for XML, so you must escape any XML delimiters, such as &.

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
```

```
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        <label>Hello world!</label>
        <help>Hello world block tooltip</help>
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>Your name</label>
          <help>Tooltip for this property</help>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Add a Help Button and Help Topic

For custom Wires blocks that require more than simple tooltips for user assistance, you can also create an HTML page as a help topic and add this to the block properties form using the macro metadata element <help> within <block>. This adds a **?** button to the properties form.

The button opens your help topic (HTML page) in a new window that is 550px wide, by default, but can be resized.

How you handle CSS and images for help topics depends on how this HTML page will be accessed:

- **Absolute URLs**: the help topic HTML page, along with any CSS stylesheets and images must be hosted in a web server or application server that is accessible to the MashZone NextGen Server. There are no special requirements for links to CSS or images.
- **Relative URLs**: the help topic HTML page and *any* images should be uploaded to MashZone NextGen by a MashZone NextGen administrator. You may also upload CSS stylesheets or simply define all classes in an internal <style> tag in the <head> element of the help topic HTML page.

  With relative URLs, you refer to CSS or images using a URL in the form `/presto/files/`*path/filename-assigned* using the path and filename assigned in MashZone NextGen.

Create the HTML page for the help topic and either:

- Deploy it, with any required resources in an appropriate web server or application server that accessible to the MashZone NextGen Server.
- Upload it, with any required resources, to MashZone NextGen.

Then add a URL to <help> in <block>:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        <label>Hello world!</label>
        <help url="http://myOrg.com/help/helloworld.html">
          Hello world block tooltip</help>
        ...
      </block>
      ...
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

This example shows an absolute URL. If you have uploaded the help topic to MashZone NextGen, use a URL in the form `/presto/files/`*path/help-topic-name* .

Make Custom Block Properties Required

To ensure that specific input parameters are supplied to a macro being used as a custom block in Wires, add the macro metadata element <required> to <parameter> within <parameters> and make the value `true`.

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>Your name</label>
          <help>Tooltip for this property</help>
          <required>true</required>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

If there are properties marked as required, Wires prevents users from running the block to preview results until all required properties are set.

Configure Field and Line Sizes for Custom Block Properties

You can make property fields be Multi-Line Input Fields. You can also set Field Width and Height for custom block properties.

Multi-Line Input Fields

By default, the block property fields for macro input parameters are a single-line entry field. You can change this to a multi-line field using the macro metadata element <type> with a datatype of `textarea` in <parameter>.

For example:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>A description</label>
          <type datatype="textarea"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Field Width and Height

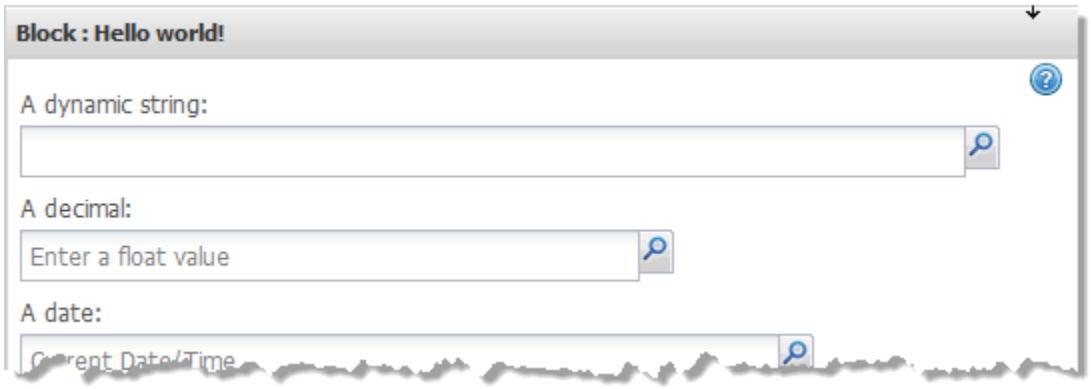
By default, Wires sets the width for block property fields to fill the width of the current Block Properties pane. See for an example.

You can use the macro metadata element <uiconfig> in <parameter> to manually set the width and height of block property fields.

Note: The <uiconfig> element contains a configuration object in JSON (Javascript Object Notation) format. Typically, you should enclose this in a CDATA section to ensure there are no conflicts between the JSON and XML syntaxes.

See "Advanced Custom Block and Property Configuration" on page 435 for more information on the options you can use with <uiconfig>.

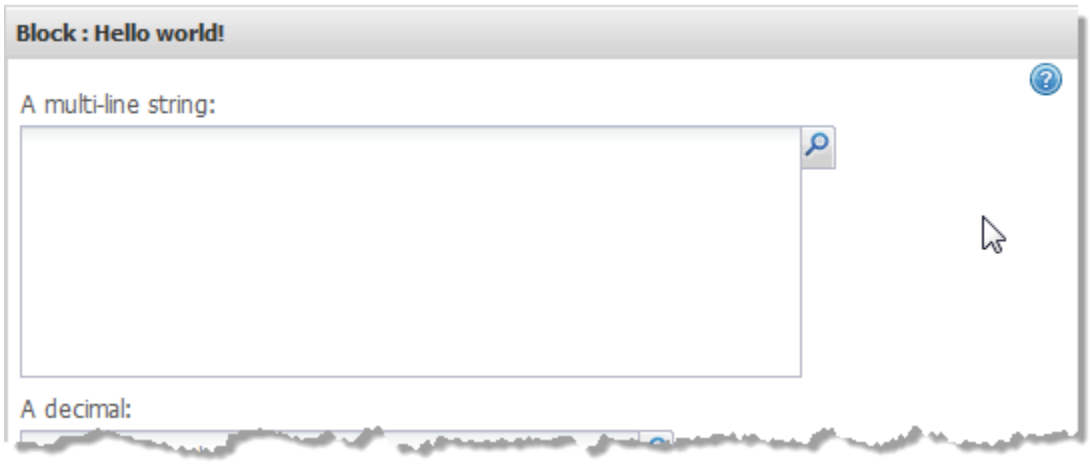
Use the `anchor` property to set the width and the `style` property to set the height. For a single-line block property, for example:



The metadata would look something like this:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        ...
        <parameter name="aNumber">
          <label>An integer</label>
          <type datatype="integer"/>
          <uiconfig>
            <![CDATA[{ "anchor":"60%" }]]>
          </uiconfig>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

For a multi-line block property, you can set both height and width:



The metadata would look something like this:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>A description</label>
          <type datatype="textarea"/>
          <uiconfig>
            <![CDATA[{ "anchor":"75%", "style":"height:120px;" }]]>
          </uiconfig>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Configure Strings, Numbers, Dates or Boolean Properties in Custom Blocks

In macros used as custom Wires blocks, you can add some specific behaviors to the block input field for macro input parameters with simple datatypes, including:

Strings	■ Make Custom Block Properties Required ■ Passwords

	■ URLs ■ Pick and Format Dates or Times ■ Configure Dynamic Path Choices and Results for Custom Blocks ■ Limit Custom Block User Entries to a List of Values
Numbers	■ Make Custom Block Properties Required ■ Integer Numbers ■ Decimal Numbers ■ Configure Dynamic Path Choices and Results for Custom Blocks ■ Limit Custom Block User Entries to a List of Values
Dates or Times	■ Make Custom Block Properties Required ■ Pick and Format Dates or Times ■ Configure Dynamic Path Choices and Results for Custom Blocks ■ Limit Custom Block User Entries to a List of Values
Boolean	■ Make Custom Block Properties Required ■ Boolean Choices

Passwords

String properties for custom Wires blocks allow user entries, by default, but show the text of the entry. To obscure the actual text of a property that is a password, you can use the macro metadata element <type> in <parameter> with a datatype of `password`.

The metadata would look something like this:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
```

```
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>A password</label>
          <type datatype="password"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

URLs

You can provide some validation for user entries in custom block properties that should be URLs with the macro metadata element <type> in <parameter> and a datatype of `url`. For example:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>A URL</label>
          <type datatype="url"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

With a `url` datatype in the macro metadata, Wires will validate that user entries begin with a valid protocol for a URL, such as `http://`.

Integer Numbers

You can force users to enter integers for custom block property that corresponds to a macro input parameter with a `number` datatype. Use the macro metadata element <type> in <parameter> and a datatype of `integer`. For example:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aNumber">
          <label>An integer</label>
          <type datatype="integer"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

With an `integer` datatype in the macro metadata, Wires does not accept a decimal point (.) or any alphabetic characters in user entries.

Decimal Numbers

You can allow users to enter decimal values for custom block property that corresponds to a macro input parameter with a `number` datatype. Use the macro metadata element <type> in <parameter> and a datatype of `decimal`. For example:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aNumber">
          <label>A decimal</label>
          <type datatype="decimal"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

With a `decimal` datatype in the macro metadata, Wires does not accept any alphabetic characters in user entries. If users enter a decimal point, they must enter at least one decimal digit in the number.

Note: This datatype does not support floating point notation for numbers.

Boolean Choices

With macro input parameters that have a boolean datatype, users must enter `true` or `false` in the corresponding property for the custom block. You can simplify user entries by having Wires provide a selection list instead.

Use the macro metadata element <type> in <parameter> with a datatype of `boolean`. For example:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aBoolean">
          <label>Is this true?</label>
          <type datatype="boolean"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

The labels in the selection list are `Yes` and `No` but the values assigned are `true` and `false` respectively.

Pick and Format Dates or Times

With macro input parameters that are dates or dates and times, you can have Wires provide a date and time control to make it easier for users to enter dates or dates and times. You can also optionally control the format of the date and time. Your choices for handling dates and times depends on the datatype you use for the macro input parameter:

- *datetime*: this is generally the best choice for an input parameter that is a date or datetime, with some limitations. This datatype supports date comparisons, sorting or XPath date math functions. However, values for dates and times *must* use the standard format:

yyyy-mm-dd `T`*hh:mm:ss[.sss zzzzzz*

> **Note:** This format is the `datetime` datatype in the XML Schema standard, which is based closely on the ISO 8601 standard.

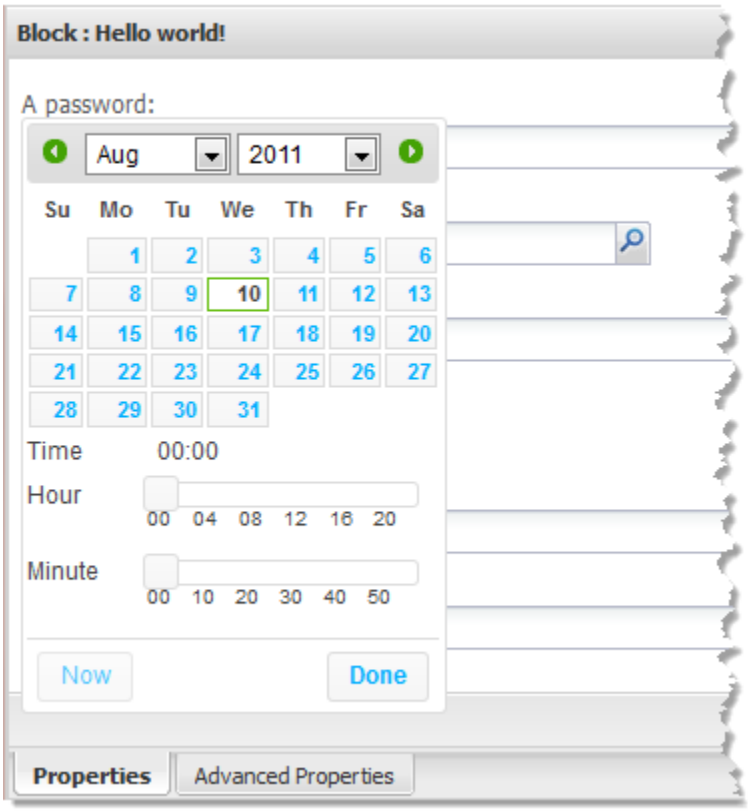
For dates in this format, years are four digits and both month and days are two digits. Times have two digit hours, minutes and seconds, using a 24-hour clock, with optional milliseconds and a GMT time zone. If no time is specified, this defaults to 00:00:00 (Midnight) and the time zone for the current user.

Dates and times are separated with a literal `T`. Times and time zone are separated by a space.

To provide a date and time control for a block property with a `datetime` datatype, add the macro metadata element <type> in <parameter> with a datatype of `datetime`:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="aDateTime" type="datetime"/>
    ...
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aDateTime">
          <label>Choose a date and time</label>
          <type datatype="datetime"/>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

The date and time control that Wires uses in this case is:



- *string*: you can also pass dates or times in input parameters using the `string` datatype. This is typically most useful when the date or time input parameter is used within the macro for the block as an input to a mashable, mashup or some other information source that requires a format that does not match the standard `datetime` format. (See previous *datetime* discussion.)

  With a `string` input parameter, you can provide a date and time control. You can also control the format that the chosen date and time uses.

  The limitation with a `string` datatype for dates and times is that it does not support date sorting, date comparisons or XPath date math functions.

  To provide a date and time control for a block property with a `string` datatype, add the macro metadata element <type> in <parameter> with a datatype of `datetime`. To define the format for the input parameter value, add the macro metadata element <datetimefield> in <type>. The metadata would look something like this:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="aStringDate" type="string"/>
    ...
    <presto:macro-meta>
```

```
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aStringDate">
          <label>Choose a future date</label>
          <type datatype="datetime">
            <datetimefield format="m/d/yy"/>
          </type>
        </parameter>
        ...
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

The `format` attribute defines a template for the format to use with this input parameter. This example would output just a date with a 1-2 digit month, 1-2 digit day and a four digit year.

The format can specify just dates or both dates and times. You can also use either a 24-hour or 12-hour clock for times.

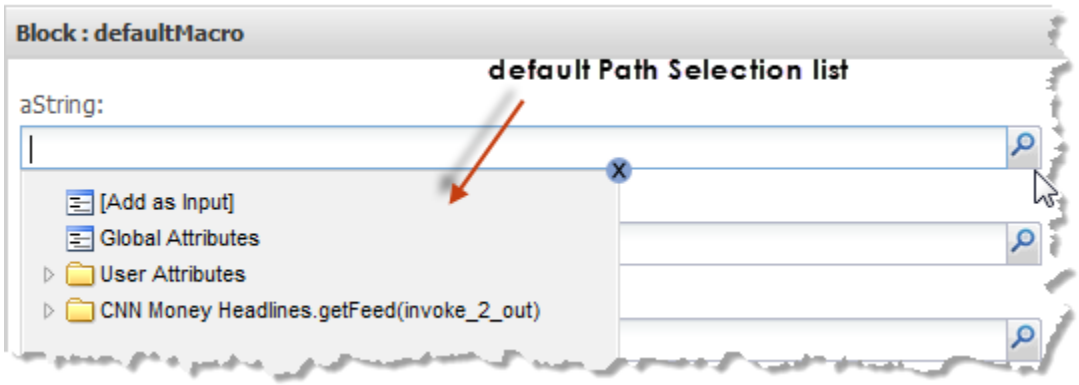
Valid format characters include:

/ or -	To separate the components in a date.
:	To separate components in a time.
space or any non-token character	Between dates and times or between times and a suffix or time zone. For example: ■ `mm/dd/yyyyThh:mm` displays a date and time such as `03/05/2012T22:06` ■ `d-m-yyyy hh:mm:ss TTZ` displays a date, time and time zone such as `5/3/2012 10:06:15 PM-0700`
y	For a two digit year.
yy	For a four digit year.
m	For a 1-2 digit month (no leading zeros).
mm (for dates)	For a two digit month (leading zeros if needed).
M	For a short month name.

MM	For a long month name
d	For a 1-2 digit day (no leading zeros).
dd	For a two digit day (leading zeros if needed).
D	For a short day of the week.
DD	For a long day of the week.
o	For 1-2 digits for the day of the year (no leading zeros).
oo	For two digits for the day of the year (leading zeros if needed).
hh	For a two digit hour (leading zeroes if needed). Times are optional. Both hour and minute are required if times are in the format.
mm (for times)	For a two digit minute (leading zeros if needed). Times are optional. Both hour and minute are required if times are in the format.
ss	For a two digit second. Seconds are optional in times.
TT or tt	If this time should be based on a 12-hour clock, this is the form of the suffix to use: AM | PM or am | pm respectively.
Z	To include a GMT time zone for times.

Configure Dynamic Path Choices and Results for Custom Blocks

By default, every property for a custom Wires block allows users to set the value for that property dynamically using the **Path Selection** list.



The default options presented to users in the **Path Selection** list include:

- Any node from the results of document-type input parameters to this custom block.
- Any node from the results of any other block in the mashup.
- Any MashZone NextGen global or user attribute.
- Any existing input parameter to the mashup.
- The option to add an input parameter to provide the value for this property.

You can change the behavior of the **Path Selection** list for individual properties of a custom block in these ways:

- Limit Dynamic Choices and Forbid Literal Entries
- Get the Chosen Path as Text

Limit Dynamic Choices and Forbid Literal Entries

You can limit the choices available within the **Path Selection** list to the results for one specific block in the mashup, and optionally to nodes within a specific *path* (a specific node or any of its descendants) in those results. This prevents users from entering literal values for the block property.

> **Note:** The results for the block that user choices should be limited to must be passed to the macro as an input parameter with a document type.

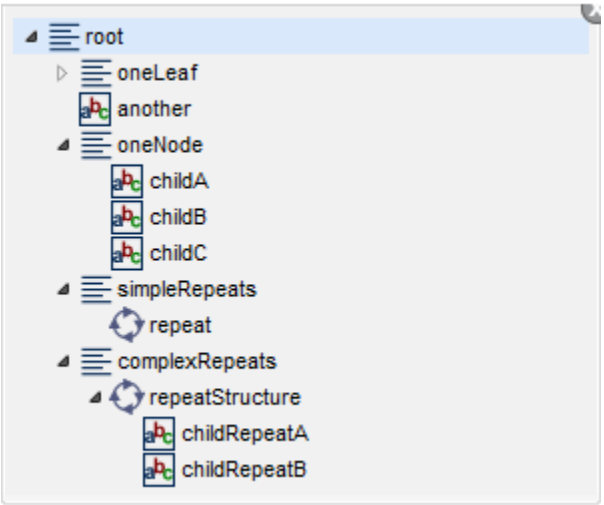
Use the macro metadata statement <type> within <parameter>. Add <xpath> within <type> and specify the input parameter that choices should be limited to in the `limitTo` attribute.

The basic metadata to limit choices to results for one block would look something like this:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
     xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
```

```
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="srcDoc" type="document"/>
    <input name="aString" type="string"/>
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="aString">
          <label>Select a leaf node</label>
          <type datatype="string">
            <xpath limitTo="$srcDoc"/>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

With this configuration, the **Path Selection** list for this example would look like this:



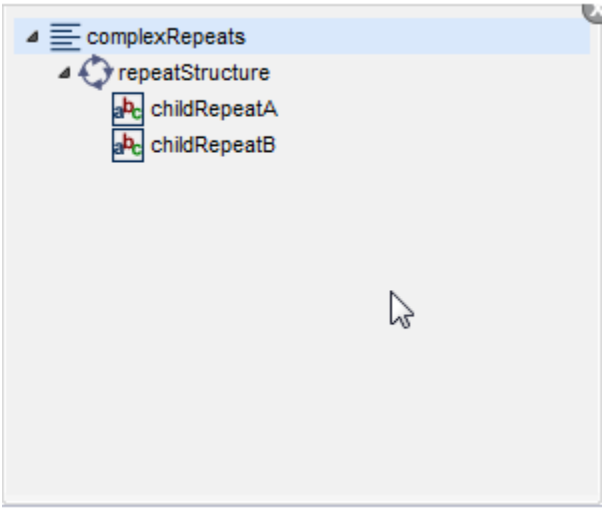
In this example, the property with this configuration is a simple type, so users would be automatically be limited to selecting leaf nodes with simple data. For a document-type property, users could select any node within this single result.

To limit choices to one path within a specific block result, add path information to the input parameter name in `limitTo` attribute. For example:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="srcDoc" type="document"/>
    <input name="aString" type="string"/>
    <presto:macro-meta>
      <block usage="Wires">
```

```
      ...
    </block>
    <parameters>
      <parameter name="aString">
        <label>Group items by</label>
        <type datatype="string">
          <xpath limitTo="$srcDoc/root/complexRepeats"/>
        </type>
      </parameter>
    </parameters>
  </presto:macro-meta>
  ....
</macro>
...
</macros>
```

With this configuration, the **Path Selection** list for this example would look something like this:



Get the Chosen Path as Text

In some cases, especially in macros that are generic, what you need for a particular block property is the XPath expression to a node that the user selects rather than actual the value of that node. Typically these expressions are used within the logic of the macro when the macro can be used with a wide variety of inputs. See "Make Macros Generic for Custom Blocks" on page 396 for more information and examples.

In this case, you want Wires to force users to pick an appropriate node from the **Path Selection** list, as discussed in "Limit Dynamic Choices and Forbid Literal Entries" on page 424. But the block property *must* be assigned the XPath expression for the selected node rather than the runtime value. To do this, the macro for a custom block must:

- Have an input parameter with a `string` datatype to hold the XPath expression.
- Have metadata for that string parameter that:
  - Includes <type>.

- Assigns a datatype of `path` to <type>.
- Includes <xpath> within <type> and uses the `limitTo` attribute to identify the document-type input parameter that contains the results that users may choose from, and optionally a specific branch within those results.
- Optionally, uses the `usage` attribute in <xpath> to define the types of nodes that users may validly select for this block property.

This next example defines a block property that accepts an XPath expression for the `stringPath` input parameter. It limits user choices for `stringPath` to any node within the results connected to the `srcDoc` input parameter:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="srcDoc" type="document"/>
    <input name="stringPath" type="string"/>
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="srcDoc">
          <label>Choose the results</label>
        </parameter>
        <parameter name="stringPath">
          <label>Choose the field to group these results</label>
          <type datatype="path">
            <xpath limitTo="$srcDoc"/>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

When `limitTo`=$*result-input-parameter-name* , the format of the XPath expression that is passed to the macro input parameter is in the form:

/root/path/to/node .

This includes the full path, starting from the root node of the selected results, but does not include the input parameter name that identifies the results.

You can further limit user choices for this XPath expression to:

- A specific branch within the selected input parameter.

  Simply add path information along with the input parameter name to the `limitTo` attribute. For example:

  ```
  <macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  ```

```
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="srcDoc" type="document"/>
    <input name="stringPath" type="string"/>
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="srcDoc">
          <label>Choose the results</label>
        </parameter>
        <parameter name="stringPath">
          <label>Choose the field to group these results</label>
          <type datatype="path">
            <xpath limitTo="$srcDoc/root/complexRepeats"/>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

When `limitTo`=$*result-input-parameter-name*/*some*/*path* , the XPath expression that is passed to the macro input parameter is in the form:

relative/*path*/*to*/*node*

This is a relative path, relative to `$`*result-input-parameter-name*/*some*/*path* .

- Specific types of nodes.

  Wires cannot predetermine what type of node users should select in the **Path Selection** list for this case. By default users may select any node within the results of the selected document-type input parameter.

  You can use the `usage` attribute on <xpath> to limit user choices to specific types of nodes:

  - `array` = users can only select repeating nodes, either simple (with data) or complex (with children)
  - `branch` = users can only select complex nodes with children, either a single node or repeating nodes
  - `leaf` = users can only select simple nodes with data, either a single node or repeating nodes
  - `path` = users can only select any node

This next example limits user choices to repeating nodes within the results connected to the `srcDoc` input parameter:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="srcDoc" type="document"/>
    <input name="stringPath" type="string"/>
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="srcDoc">
          <label>Choose the results</label>
        </parameter>
        <parameter name="stringPath">
          <label>Choose items to iterate over</label>
          <type datatype="path">
            <xpath limitTo="$srcDoc" usage="array"/>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Limit Custom Block User Entries to a List of Values

Another common requirement for custom Wires blocks is to limit user entries for input parameters with simple datatypes to a specific list of valid values.

You can define the list as literal values using <type> and <list> within <parameter>. See "Literal List Values" on page 429 for instructions.

You can also supply the list of values dynamically using two techniques: Dynamic List Values from a Macro Input Parameter or Dynamic List Values from a Known Source.

Literal List Values

To provide a list of literal valid values, you add the macro metadata element <type> to the corresponding <parameter> and set the datatype to `enum`. Add <list> as a child of <type>:

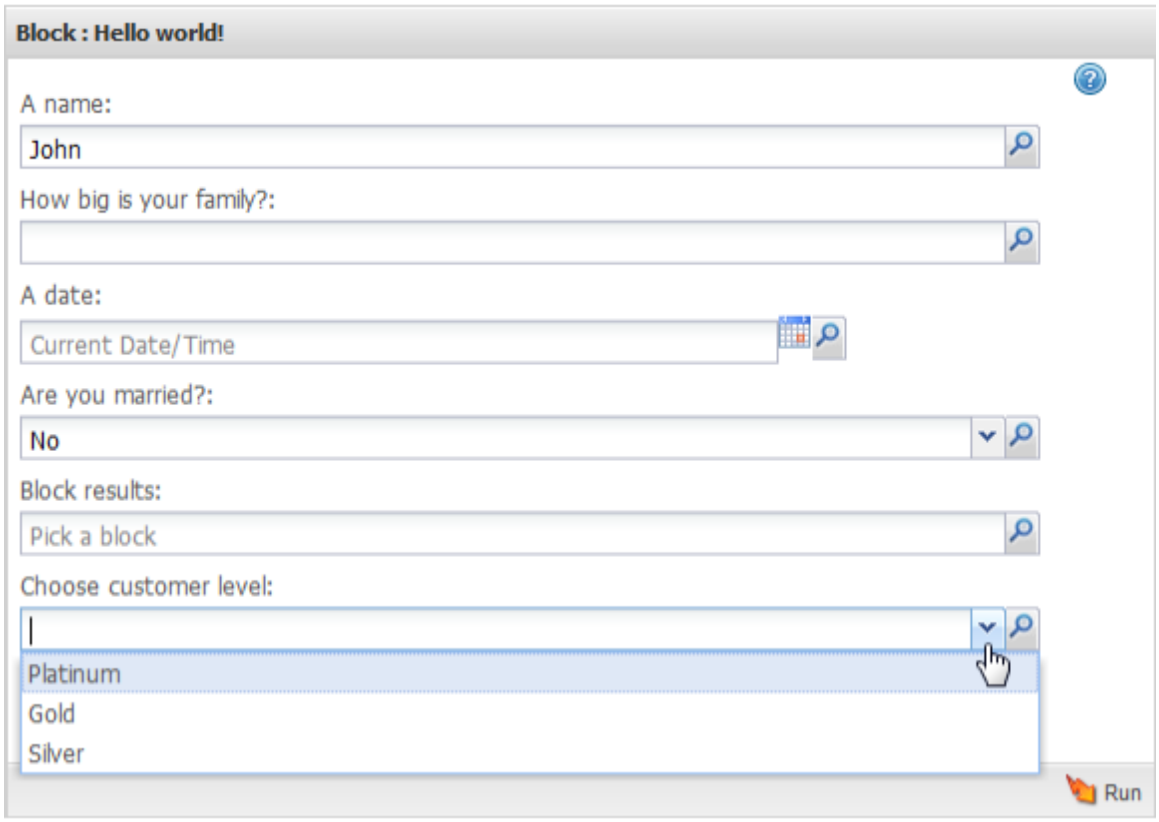
```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
```

```
        ...
        <parameter name="aList">
          <label>Choose a customer category</label>
          <required>true</required>
          <type datatype="enum">
            <list></list>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

You then define the valid values as a simple string with values separated by commas using the <values> element inside <list>. With this configuration, the values you supply are also the labels that users see in the list:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        ...
        <parameter name="aList">
          <label>Choose a customer category</label>
          <required>true</required>
          <type datatype="enum">
            <list>
              <values>platinum,gold,silver</values>
            </list>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

If the values are not self-explanatory or you simply want to provide alternate labels, use a set of <option> elements rather than <values>.



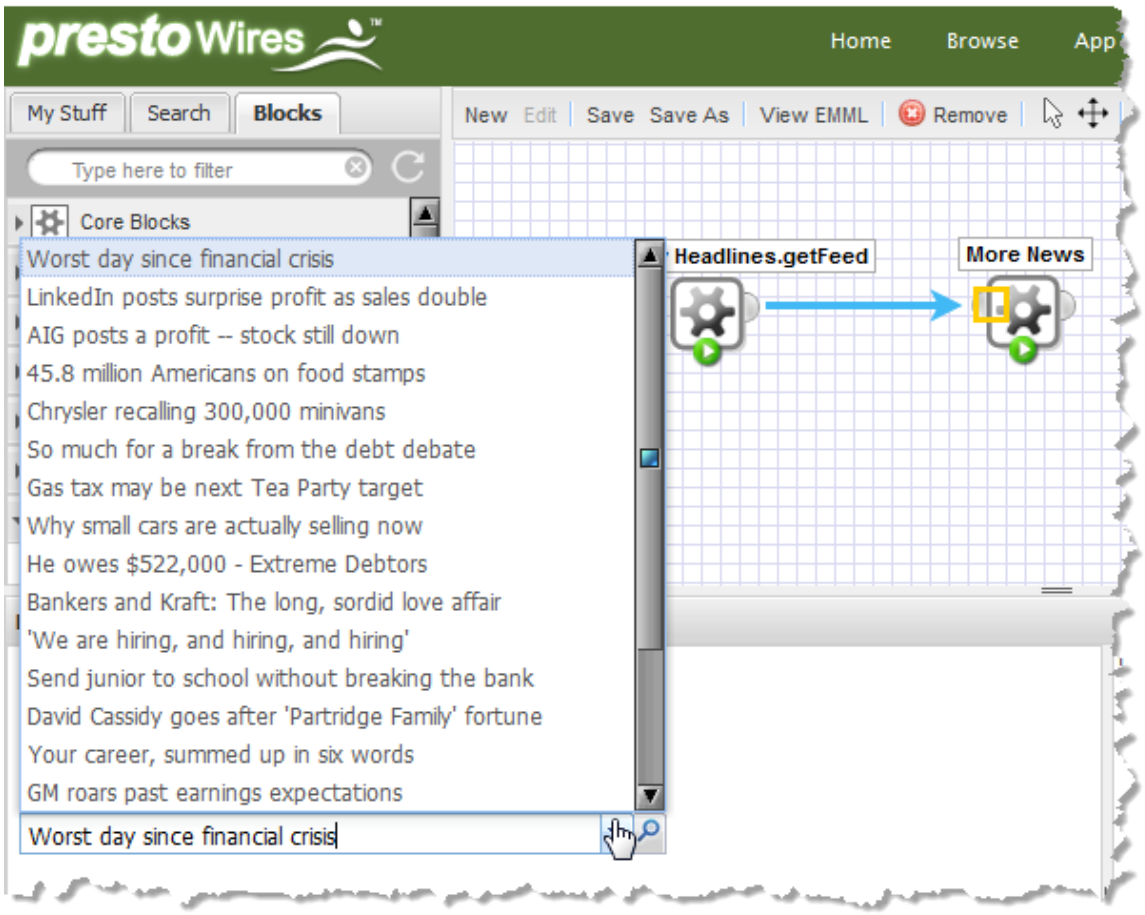
Define the label to show in the `label` attribute and the value to send to the macro as the value of <option>:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        ...
        <parameter name="aList">
          <label>Choose a customer category</label>
          <required>true</required>
          <type datatype="enum">
            <list>
              <option label="Platinum">1</option>
              <option label="Gold">2</option>
              <option label="Silver">3</option>
            </list>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
```

```
    ....
  </macro>
...
</macros>
```

Dynamic List Values from a Macro Input Parameter

You can populate the list of valid values dynamically from another input parameter to the macro. For example, one input parameter is a document with results from an RSS web feed. A second parameter accepts one article title that users should select from the RSS results to use as a query to another mashable, such as this example:



Thus one field in the repeating items from the first input parameter provides the dynamic list of valid values for a second macro input parameter. The first input parameter must be a document type that has at least one repeating node.

To provide a dynamic list of values for an input parameter based on a repeating node from another parameter, you add the following macro metadata elements to <parameter>:

- `<type>` with a datatype of `enum`
- `<list>` as a child of <type>

- `<xpath>` as a child of <list>
- You specify the path within the source input parameter to the repeating node that contains the values for this list in the `limitTo` attribute in <xpath>

In this example, the <item> node in the RSS feed for the newsSource input parameter is repeating. The list of values to show for the newsItem input parameter is the article titles for each <item> in newsSource:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="MoreNews"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <input name="newsSource" type="document"/>
    <input name="newsItem" type="string"/>
    <presto:macro-meta>
      <block usage="Wires">
        ...
      </block>
      <parameters>
        <parameter name="newsSource">
          <label>Choose an RSS web feed</label>
          <required>true</required>
        </parameter>
        <parameter name="newsItem">
          <label>Choose a news article</label>
          <required>true</required>
          <type datatype="enum">
            <list>
              <xpath limitTo="$newsSource/rss/channel/item/title"/>
            </list>
          </type>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

Dynamic List Values from a Known Source

To provide a dynamic list of valid values from mashables, mashups or other information sources, you add the macro metadata element <type> to the corresponding <parameter> and set the datatype to `enum`. You invoke the mashables, mashup or other source of information to obtain the dynamic list using <uiconfig>:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
```

```
      <block usage="Wires">
        ...
      </block>
      <parameters>
        ...
        <parameter name="aList">
          <label>Choose a country</label>
          <type datatype="enum"/>
          <uiconfig><[CDATA[ ... ]]></uiconfig>
        </parameter>
      </parameters>
    </presto:macro-meta>
    ....
  </macro>
...
</macros>
```

The contents of <uiconfig> is a JSON object with configuration for this property. It is a best practice to wrap the <uiconfig> content in a CDATA section, as shown in this example.

To get a dynamic list of values, the configuration object must include a property named `acceptedValues`. This property must contain an array with the list of values to display.

To obtain this list dynamically, you define a *self-invoking, anonymous* function as the value of `acceptedValues`. The function is invoked immediately after it is defined, during rendering of the custom block properties form, and returns the array of values to display.

This example uses MashZone NextGen Connect for JavaScript (PC4JS) to invoke a MashZone NextGen mashable named `CountryCodes` and build the array of valid values for a block property from the results:

```
<uiconfig><![CDATA[{acceptedValues: (function(){
      var countries = [];
      var connection=new Presto.Connection('/presto');
      connection.request( {
          url: "/presto/edge/api/rest/CountryCodes/getData?x-presto-resultFormat=json",
          type: "get",
          contentType: "application/x-www-form-urlencoded",
          data: "" },
        { onSuccess: function(response, responseHeaders) {
            var result=response;
            list = result["e:DataTable"]["e:Entry"];
            for (var i=0; i < list.length; i +=1) {
              var country = list[i].Name;
              countries.push(country);
            } },
          onFailure: function(e){
            console.log("service call failed");
            } } );
      return countries;
    })()
  } ]]></uiconfig>
```

Some points to keep in mind:

- See "Use MashZone NextGen Connect for Javascript" on page 1484 for basic information on using PC4JS.

- You can find the URL and any other technical information needed to invoke a mashable or mashup in Technical Specs on the artifact page for that mashable or mashup. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information.
- When users add the custom block to a mashup in Wires, the list of values for the block property can only be successfully populated if that user has permission to run the mashable or mashup that is being invoked. Make sure that the run permissions for this mashable or mashup are set appropriately.
- Because the results from the mashable used in this example have a namespace, the code to loop through the results to populate the array uses a `namespace-prefix:node-name` syntax. The specific prefix to use for the namespace is visible in the Tree View for results for the mashable or mashup.
- You can also use jQuery functions to send AJAX requests to retrieve the dynamic list values.

Advanced Custom Block and Property Configuration

The macro metadata element <uiconfig> provides several ways to further customize aspects of a custom Wires block or of individual property fields for custom blocks. This metadata element uses JSON configuration objects for various classes in Ext JS, plus additional configuration properties that are specific to MashZone NextGen.

Important: When <uiconfig> is used to completely define one block property or all block properties, you *must* meet the licensing terms for Ext JS as this use is not covered by your MashZone NextGen license.

The techniques discussed in this topic are advanced configuration using JavaScript and Ext JS. Users should be comfortable with the Ext JS 4.0 Library and JavaScript programming in general.

With <uiconfig>, you can set:

- Additional Configuration for Blocks
- Configure Field and Line Sizes for Custom Block Properties for individual block property fields.
- Dynamic List Values from a Known Source for block property fields that are limited to a list of valid values.
- Additional Configuration for Block Property Fields

Note: For information on the Ext JS configuration properties that you may use in <uiconfig> elements, see Ext JS documentation at http://docs.sencha.com/ext-js/4-0/.

Additional Configuration for Blocks

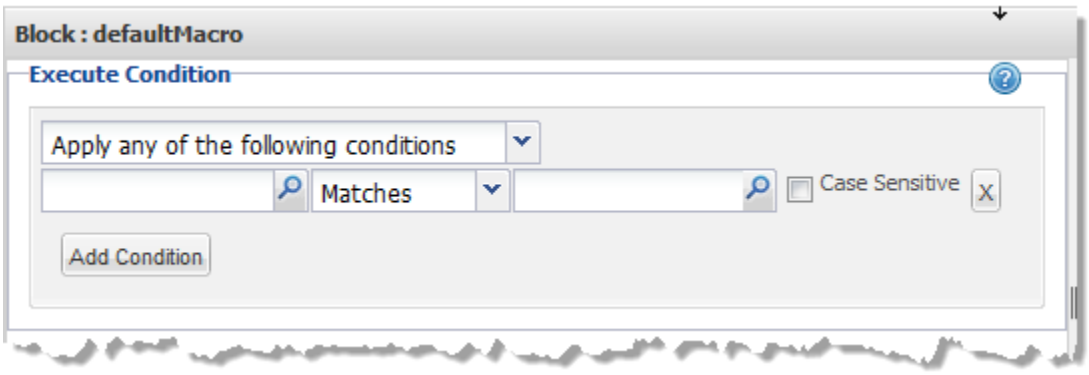
You may use <uiconfig> as a child of <block> to provide configuration for the entire block in addition to configuration you have defined in <block> and <parameters>. You define a JSON configuration object in <uiconfig>.

> **Note:** It is a good practice to enclose the JSON code for <uiconfig> in a CDATA section to ensure there are no conflicts between the JSON and XML syntaxes.

In this context, the JSON object can include any configuration properties for the `Ext.form.FormPanel` class plus these additional properties specific to MashZone NextGen:

- `conditionalExecution` = `true` or `false`.

  This MashZone NextGen block property is true by default, which allows the **Execute Condition** section to appear in the **Advanced Properties** tab of a custom block.



  Execute conditions allow users to only conditionally have blocks run based on one or more conditions that must be met.

- `description` = a tooltip for the block in the Blocks Menu. This is identical to the <help> metadata element within <block>, although this does not support a help button and help topic.
- `label` = the label to appear in the Block Menu for this block. This is identical to the <label> metadata element within <block>.
- `properties` = defines all the properties for the custom block when <uiconfig> has a `uiProvider`. This is only appropriate when you wish to have complete control of all aspects of the block properties form for a custom Wires block.
- `style` = to specify custom icons to use in the Blocks Menu and the Wires canvas for a custom block. This is equivalent to the <icon> metadata within <block>.

  This property is an object with the following properties to specify each icon:

  - `graph` = the URL to the custom icon for the Wires canvas. This is equivalent to <icon usage="canvas">.

- `sidebar` = the URL to the custom icon for the Blocks menu in Wires. This is equivalent to <icon usage="menu">.

See "Block Icons" on page 407 for more information on icon sizes and visual requirements.

- `uiProvider` = a subclass of `Ext.form.FormPanel` to use to take complete control over the block properties for this custom block. If you use this configuration property, you must specify the properties for this block in the `properties` configuration property within this JSON object. This completely overrides any other metadata configuration defined in <block> or in <parameters>.

The metadata to prevent conditional executions, for example, would look something like this:

```
<macros xmlns="http://www.open-mashup.org/schemas/v1.0/EMML"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/..
/schemas/EMMLPrestoSpec.xsd"
    domain="myBlocks">
  <macro name="helloWorld"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
    <presto:macro-meta>
      <block usage="Wires">
        <label>Hello world!</label>
        <help>Tooltip for this block</help>
        <icon usage="menu" url="http://myOrg.com/images/smallMacroIcon.png"/>
        <icon usage="canvas" url="http://myOrg.com/images/lgMacroIcon.png"/>
        <uiconfig><![CDATA[{"conditionalExecution":false}]]></uiconfig>
      </block>
 ...
  </presto:macro-meta>
```

The Block Properties form uses the Ext `FormLayout` which is based on the `AnchorLayout`.

Additional Configuration for Block Property Fields

You may use <uiconfig> as a child of <parameter> to provide additional configuration for one block property field in a custom block. You define a JSON configuration object in <uiconfig>.

Note: It is a good practice to enclose the JSON code for <uiconfig> in a CDATA section to ensure there are no conflicts between the JSON and XML syntaxes.

In this context, the JSON object can include any configuration properties for the `Ext.form.Field` class, or a subclass you have specified, plus these additional properties specific to MashZone NextGen:

- `acceptedValues` = an array of valid values for this field. Use this *only* when the datatype in <type> is `enum`. See "Dynamic List Values from a Known Source" on page 433 for more information and an example.

- `datatype` = the datatype for this field. This is identical to datatype for <type> and may have the same values. See "<type>" on page 588 for details.
- `doNotStripNS` = `true` or `false`. This is only applicable for block properties that return XPath expressions for a node that users select.

  It is true by default, which returns XPath expressions that include a wildcard for potential namespaces for each node in the path. For example:

  ```
  /*:root/*:some-node
  ```

  If you need to obtain an XPath expression that does not include namespaces, set this property to false.
- `label` = the label for this property field. This is identical to <label> metadata within <parameter>.
- `required` = `true` or `false`.

  This determines whether Wires treats this property as required for this custom block. It is identical to <required> metadata within <parameter>.
- `xpathPicker` = a JSON object with additional properties that configure the **Path Selection** list for this block property. Configuration properties for the **Path Selection** list include:
  - `combineWithUniversalPicker` = `true` or `false`. This defaults to false. If true, the **Path Selection** list shows both this customized list and the default **Path Selection** list.
  - `height` = the number of pixels for the height of this list.
  - `ignorePathToNode` = `true` or `false`. This defaults to false. If true, this returns the value of the selected node rather than the path to the selected node.
  - `neverHide` = `true` or `false`. This defaults to false. If true, the **Path Selection** list remains open until users explicitly close it. This is typically used to allow users to select multiple nodes.
  - `pathType` = how namespaces should be handled in the XPath expression from the **Path Selection** list:
    - `path` = uses a wildcard character as the namespace for nodes in the XPath expression. This is the default.
    - `pathWithActualNS` = uses the actual namespace, if any, for nodes in the XPath expression.
    - `pathWithoutNS` = removes any namespaces for nodes in the XPath expression.
  - `showOnlySelectablePaths` = `true` or `false`. This defaults to false. If true, it causes the **Path Selection** list to show only those nodes that are valid to select. Typically, valid selections are determined by the datatype for the input parameter for this block property.

- `xpathPreferences` = a JSON object with the following properties that identify what types of nodes are valid for user choices in the **Path Selection** list. Each property may be `true` or `false`:
  - `acceptArrayLeafNodes` = repeating nodes with data
  - `acceptArrayNodes` = repeating nodes with children
  - `acceptComplexNodes` = any node
  - `acceptLeafNodes` = nodes with data

6 Mashups in EMML

- Writing Mashups in EMML 442
- Migrating Mashups to a New EMML Version 594
- Expressions for Mashups 596
- Advanced Mashup Techniques 606
- Mashup Samples 659
- Mashup Utilities 664
- EMML Reference 670

MashZone NextGen Wires allows both power users and developers to create mashups in a simple, graphical way. The features and scope for Wires mashups, however, is only part of the full capabilities and flexibility possible with mashups.

Developers who need more robust mashup capabilities can create mashups and macros using the Enterprise Mashup Markup Language (EMML). This XML vocabulary is a very simple, but very powerful way to quickly build mashups and macros from MashZone NextGen mashable information sources, information sources accessible by URL, databases, snapshots or other information sources supported by MashZone NextGen Analytics.

Developers can use the MashZone NextGenMashup Editor to write mashups in EMML.

See "Creating a Mashup Script with EMML" on page 443 for the steps involved in writing mashups with EMML. This sections also covers all of EMML's syntax. EMML also uses XPath and other types of expressions. See "Expressions for Mashups" on page 596 for more information. See "Advanced Mashup Techniques" on page 606 for discussions of some of the intermediate and advanced techniques you can use in mashups.

Once you have a mashup script, you can Test Mashup Scripts or Macros in Mashup Editor and then save and Publish a Mashup Script in the Mashup Editor the mashup. Other tasks or examples that you may find useful include:

- Running Mashables or Mashups and Other Tasks
- Mashup Samples
- "Mashup Utilities" on page 664

Writing Mashups in EMML

Mashups allow you to easily combine or transform content from one or more mashable information sources or other mashups to produce information for situational applications, ad hoc applications that you can quickly tailor to fit your current needs, apps or dashboards. You use mashup scripts to easily define how different information sources, *mashables*, large datasets and other mashups, should be combined and transformed to produce just the information you need.

You can also use mashups:

- As components within other mashups, combining and building on your work.
- To provide pre- or post-processing for mashable or other information sources .
- To virtualize mashables, providing a simpler interface for business users to work with.
- As generic templates that can be applied to many different mashables for common patterns.

- To retrieve, query and analyze large datasets using the Real-Time Analytics Query Language.

For some simple examples of mashups and information on the Mashup Samples project, see "Mashup Samples" on page 659.

Creating, Testing, Saving and Publishing Mashups

To define mashups and make them available to other users

1. Start a mashup script in Mashup Editor or with any XML editor. See "Creating a Mashup Script with EMML" on page 443 for instructions.

   Use EMML statements to get information from sources and work with that information to produce the final data you are intersted in. See "Creating a Mashup Script with EMML" on page 443 for information and links to the basic EMML statements and techniques for mashups.

   For more advanced mashup techniques, see "Advanced Mashup Techniques" on page 606.

2. Test or debug your script and update it as needed until you are satisfied with the results for all of its operations. See "Test Mashup Scripts or Macros in Mashup Editor" on page 683 or "Testing Mashup Scripts from the Command Line" on page 665 for instructions.

3. Once the script is working correctly, publish it as a mashup in MashZone NextGen. See "Publish a Mashup Script in the Mashup Editor" on page 684 or "Publishing Mashup Scripts from a Command Line" on page 666 for instructions.

You can also update your mashup scripts once they are published (only the owner or MashZone NextGen administrators may update a mashup). Change the script as needed and republish the script using the same credentials you originally used to publish it. Or see "Updating Mashup Scripts from the Command Line" on page 667 for instructions.

Creating a Mashup Script with EMML

A mashup script is an XML file that uses the Enterprise Mashup Markup Language (EMML). They define the mashable information sources and other information sources to be used by a mashup and the actions to apply to this data to construct the results of the mashup.

Mashup scripts can be very simple, invoking a single mashable information source and filtering the output for example. They also support virtually any level of complexity.

This task discusses how to write a mashup script in EMML using the Mashup Editor. If your MashZone NextGen license supports this, you can also create mashups that use EMML extensions for the Real-Time Analytics Query Language.

You can also use any XML editor with the EMML schema in *MashZoneNG-install* /mashupclient/schema/EMMLSpec.xsd. If you need to use MashZone NextGen extension

elements, such as <presto:presto-meta>, you should use the MashZone NextGen EMML Extensions schema in *MashZoneNG-install* /mashupclient/schema/EMMLPrestoSpec.xsd.

To create a mashup script using EMML

1. Start a new mashup in the Mashup Editor:

   a. Select **Mashups >** Mashup Editor in the MashZone NextGen Hub main menu.

      The Mashup Editor opens with a new mashup script already started. If the Mashup Editor is already open, simply click **New** to open a new tab for a new mashup.

   b. Change the **name** in the <mashup> tag to the logical name for this mashup.

      Commonly, this matches the mashup's name in MashZone NextGen, although that is not required.

      Note: MashZone NextGen uses the mashup name to assign a unique identifier to the mashup. Mashup names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

2. Add EMML elements directly to invoke mashables or mashups, call macros, define variables and perform actions to define the mashup result. Use the **Mashables**, **Mashups**, **Macros** and **Actions** menus to add EMML elements. Declare any namespaces that are used in the script on the <mashup> element.

   The elements that you add to mashups are either:

   - *Declarations* for variables, parameters, data sources, namespaces, macros or metadata.
   - *Statements* that perform actions for the mashup. This includes statements to invoke mashables or other mashups, control flow, act on the results or use custom statements defined in macros.

For	Use These Elements or Attributes
Declaring Mashup and Macro Variables and Parameters	■ <variables>: contains one or more <variable> definitions to hold the input or output for any mashup script statement. ■ <input>: defines an input parameter for this mashup. ■ <output>: defines the name and type for the result of this mashup. Previous versions of EMML also used <inputparam> and <outputparam>. These declarations are *deprecated*.
Declaring Namespaces	*xmlns attribute on <mashup> or <macro>*: declares a namespace used by:

For	Use These Elements or Attributes
	■ A mashable information source called in the mashup. ■ Macros called in the mashup. See Calling a Macro for more information. ■ MashZone NextGen extension statements. See EMML Namespaces for the namespace you must declare when you use MashZone NextGen extensions such as <presto:presto-meta>. ■ Parameters or variables constructed with literal XML. The <namespaces> element is *deprecated*.
Declaring Data Sources	<datasource>: defines connection information to one database for use with direct SQL statements in this mashup. You can also simply used named datasources defined in MashZone NextGen.
Invoking Component Mashups and Mashables	■ <invoke>: invokes the operation for a published MashZone NextGen mashable information source. See also Automatically Generating <invoke> Statements in Mashup Editor. ■ <directinvoke>: Invokes an ungoverned, publically accessible web service or web site. Only HTML, Syndicated Feeds, REST web services and SOAP web services are supported. See also Controlling Component Mashup/Mashable Invocation at Runtime, Wrapping POJO Classes with Mashups and Web Clipping with a Mashup Script.
Issuing SQL Statements Directly to a Datasource	■ <sql>: to execute SQL queries to a datasource. ■ <sqlUpdate>: to execute any other SQL statement to a datasource. ■ <sqlcall>: to execute a stored procedure for a datasource. See also Handling or Throwing Exceptions.
Transforming Intermediate Results	■ <assign>: copies, and optionally transforms, one variable or variable fragment to another variable. This uses XPath 2.0 expressions to identify the source nodes to copy and optionally applies XPath functions to transform the result. This statement can also be used to assign literal values.

For	Use These Elements or Attributes
	■ <filter>: filters a set of nodes in a variable based on a filter expression in XPath 2.0. ■ <group>: find unique values, and optionally filter, a set of repeating nodes in a variable. This constructs a result document based on those groups. This statement supports multiple levels of grouping and calculations on groups. ■ <sort>: sorts a set of nodes in a variable based on sort keys and a sorting expression in XPath 2.0. ■ <annotate>: adds attributes or children elements to a selected node of a variable and assigns values using an XPath expression. This is typically used to add data to the mashup result or an intermediate variable. ■ <script>: defines a script to execute at runtime at the specified location in mashup processing. You can include JavaScript scripting code directly or point to an external file with any supported scripting language on the local server. The MashZone NextGen Server supports JavaScript and Groovy as scripting languages inside mashup scripts. Scripting can also access any Java class in the classpath. ■ <xslt>: process an input document using an XSLT stylesheet to tranform and construct a result document. See also Handling or Throwing Exceptions.
Combining Component Mashable or Mashup Results	■ <join>: joins the results of two or more mashable information sources or other mashups based on a join condition. This element works like a database join, where the results may be disparate but must have key nodes that determine how data is joined. You can also define the structure and nodes to include in the result of the join. ■ <merge>: merges the results of two or more mashable information sources or other mashups that have homogenous result models. The element works like a database union. The results of all services must have identical structures.
Constructing the Mashup Result, Inputs or Intermediate Variables	■ <constructor>: creates a structure for the result of this mashup or for an intermediate variable. You define the nodes of the result and use mashup expressions to define the data from a variable to fill these nodes.

For	Use These Elements or Attributes
	■ <appendresult>: appends a structure of nodes to the result of this mashup or to an intermediate variable. This is typically used in repeating loops to handle results with repeated sections. You define the nodes of the structure to append and use mashup expressions to define the data from a variable to fill these nodes. ■ <select>: creates a structure for the mashup result or for an intermediate variable with only selected nodes from a repeating set of items. See also <join> with a <select> child and <group>.
Controlling Mashup Processing Flow	■ <if>: handles if-elseif-else processing for a mashup script based on a condition defined in XPath 2.0. You can include any mashup statements in any section of <if>. ■ <for>: processes a loop of any mashup statements based on a sequential count. ■ <foreach>: processes a loop of any mashup statements either iterating sequentially through each node in a set of nodes from a variable or processing each loop concurrently. The set is an XPath 2.0 'sequence' defined by an expression. ■ <while>: processes a loop of any mashup statements as long as a condition is true. The condition is defined in a XPath expression. ■ <break>: explicitly stops processing the current loop and all subsequent loops in <for>, <foreach> or <while>. This can be used in <if>, <elseif> or <else> when they are used within looping statements. ■ <parallel>: processes groups of mashup statements concurrently, as separate parallel flows. The groups of statements for each concurrent task are defined within <sequence> children, which can include variable declarations and any mashup statement. ■ Database Mashable Transactions: using <presto:beginTransaction>, <presto:commitTransaction> and <presto:rollbackTransaction>. ■ SQL Transactions: using <sqlBeginTransaction>, <sqlCommit> and <sqlRollback>.

For	Use These Elements or Attributes
Handling or Throwing Exceptions	■ <try>: to execute a block of EMML statements and catch and handle any exceptions from those statements. ■ <throw>: to throw an exception for this mashup or macro. If this exception is not caught in a <catch> block in this mashup or macro, this also forceably stops mashup/macro processing. ■ <return>: to forceably stop mashup or macro processing and return the current value of <output> as the mashup or macro result.
Defining and Using Custom Mashup Statements with Macros	■ <macro>: defines a custom EMML statement for use in mashups. Macros can contain any mashup statements. They accept input parameters and can return a result in an output variable. ■ <macros>: root node for a macro library containing one or more macro definitions for use in any mashup hosted by a MashZone NextGen Server. See also Creating and Registering Macros. ■ Including Macro Domains in Mashup Scripts or Macros using <include>. ■ Calling a Macro using the custom statement <macro:custom-macro-name> in a mashup script. See also Adding Custom Blocks to MashZone NextGen Wires Using Macros.
Supporting Debugging	■ <display>: sends debugging messages to the log and console, optionally including the value of a variable or a portion of a variable. ■ <assert>: defines assertions that are evaluated at runtime. Failures throw exceptions.
Adding Metadata to Mashups	■ <emml-meta> for metadata built into EMML ■ <presto:macro-meta> for metadata for macros that are used as custom blocks in Wires or used solely in mashups written in EMML ■ <presto:presto-meta> for metadata built into MashZone NextGen ■ <user-meta> for custom metadata to meet your requirements

For	Use These Elements or Attributes
Working with large datasets using RAQL	See Getting Started with MashZone NextGen Analytics and RAQL Queries for more information on the EMML extensions you can use with large datasets and RAQL.

For example:

```
<mashup name="NewsStories"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
      ../xsd/EMMLSpec.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
  <variables>
    <variable name="stories" type="document"/>
  </variables>
  <output name="result" type="document"/>
  <invoke service="YahooRSSFeed" operation="getFeed"
      outputvariable="stories"/>
  <filter inputvariable="stories"
      filterexpr="$stories/rss/channel/item[matches(description,'Business')]"
      outputvariable="result"/>
</mashup>
```

3. Once the mashup is complete:
   a. Test the mashup. See Test Mashup Scripts or Macros in Mashup Editor for instructions.
   b. Click **Save As** to save, and optionally publish, your mashup script:
      - Enter a **Name** for the mashup. MashZone NextGen uses the mashup name to assign a unique identifier to the mashup. Mashup names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`
      - Optionally, enter a **Description** for the mashup.
      - Optionally, enter **Tags** or select the **Provider** for the mashup.
      - If the mashup is not yet ready for other users, clear the **Publish** option. Set this option to turn the mashup on and allow other users to work with it.
      - Click **Save**.

<mashup>

A script defining a mashup that may combine, filter or otherwise process the results from one or more published mashable information sources or publicly accessible services (component services including other mashups) or web sites.

For more information and examples, see "Creating a Mashup Script with EMML" on page 443.

Can Contain	operation | (Declarations Group | Variables Group | presto:presto-meta | Statements Group | presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction | Macroincludes Group | macro:custom-macro-name | (any element in a non-EMML namespace)*)+
Allowed In	Root element with no parents.
Namespaces	EMML Namespaces

Attributes

Name	Required	Description
name		The logical name for this mashup. Names must be unique and can contain characters from the character sets supported by the MashZone NextGen Repository, numerals, underscores (_) or dashes (-). Names must start with a letter. If omitted, the name for this mashup service must be supplied when the service is published with MashZone NextGen.

EMML Namespaces

The Enterprise Mashup Markup Language uses a distinct namespace for updated versions of its schema. In many cases, the language is backwards compatible, mashup scripts from previous versions still work as expected.

MashZone NextGenalso defines extensions to EMMLwhich use the MashZone NextGenMashup Extension namespace. Custom statements that you define for your mashups as EMML macros use the macro reference namespace.

The EMML, MashZone NextGen extension and macro namespaces that are currently supported include:

EMML Mashup Namespace	http://www.openmashup.org/schemas/v1.0/EMML
	http://www.jackbe.com/2008-03-01/EMMLSchema

	Important: Deprecated. Please upgrade to a supported namespace.
	http://www.jackbe.com/2007-09-15/JMMLSchema **Important:** Deprecated. Please upgrade to a supported namespace.
EMML Macro Reference Namespace	http://www.openmashup.org/schemas/v1.0/EMMLMacro
	http://www.jackbe.com/2008-03-01/EMMLMacro **Important:** Deprecated. Please upgrade to a supported namespace.
MashZone NextGen Mashup Extension Namespaces	http://www.jackbe.com/v1.0/EMMLPrestoExtensions

Mashup scripts or macro libraries from earlier versions and namespaces may no longer function and should be migrated to the current version of the schema.

<operation>

The named operation for this mashup. If omitted, the mashup script defines an operation with a default name of "Invoke".

Can Contain	(Declarations Group | (Variables Group) | (presto:presto-meta) | Statements Group | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | Macroincludes Group | ((macro:custom-macro-name | any element in a non-EMML namespace)*))+
Allowed In	mashup

Attributes

Name	Required	Description
name		The optional name for this operation. This defaults to `Invoke`.

Declaring Mashup and Macro Variables and Parameters

Mashups and macros have one output parameter to hold their result. You can also declare any number of input parameters to pass data to mashups or macros.

Note: Only one <output> declaration can occur in a mashup or macro. Validation for mashup scripts or macro libraries will *not* flag multiple <output> declarations as an error. You will, however, receive an error when compiling, debugging or publishing the mashup script.

You can also declare variables in mashups, macros or other control statements that allow nested EMML statements in order to work with data in the mashup or macro. For example, you can declare variables within <foreach>.

You *must* declare parameters explicitly. With variables, you can declare them explicitly or you can create variables implicitly by assigning a name to the `outputvariable` attribute for any statement. Explicitly declaring variables does make mashup scripts easier to understand and also makes content assistance available. Variables and parameters *must* be declared (explicitly or implicitly) before they are used in any other EMML statement.

Important: In previous versions of MashZone NextGen, variables and parameters were *not* required to be declared before they were used.

Variables and parameters all have a name, a datatype and optional subtype, data (a value) and a scope. They may also have a default value. Input and output parameters can also have a label, for use in user interfaces. For more information, see:

- "Valid Names for Variables and Parameters" on page 452
- "Datatypes for Variables and Parameters" on page 453
- "Default and Constructed Values" on page 454
- "Variable and Parameter Scopes" on page 455
- "Referencing Parameters and Variables" on page 457
- "Wrapping Results or Variables in CDATA Sections" on page 619
- "Handling HTML Responses" on page 623
- "Handling JSON Responses or Inputs" on page 624
- "Working Samples" on page 457

Valid Names for Variables and Parameters

Variable and parameter names must follow these rules to be valid:

- They cannot use the following reserved names:
  - *fault*

- *faultcode*
- *faultexception*
- *faultmessage*

These reserved names are used for mashable invocation error handling.

- Any *JavaScript, Java or XML* reserved keyword. For example:
  - `document` is invalid as this is a reserved keyword in JavaScript.
  - `xml` is invalid because it is reserved in XML.
  - `this` is invalid because it is commonly used in both Java and JavaScript.
- They can use these reserved names if they refer to MashZone NextGen attributes:
  - *user*
  - *session*
  - *global* or *system*

  For examples and more information, see "Using MashZone NextGen Attributes in Mashups" on page 614.
- They can use the reserved name *httpResponse.header-name* to define standard or custom HTTP headers for the mashup result. For custom HTTP headers, *header-name* must begin with `x-`.
- They must be unique within the scope that they are declared in. However, local variables can use the same name as global variables. See "Valid Names for Variables and Parameters" on page 452 for more information.
- They must start with an ASCII letter.
- They can contain ASCII letters, numbers, dashes (-) or underscores (_). Do *not* use any other symbols or punctuation.

Datatypes for Variables and Parameters

Variable and parameter data can be scalar values or complex objects:

Scalar Types	`string`
	`number`
	`date` (*deprecated*, use `datetime` instead)
	`datetime`
	`boolean`

Complex Types (objects)	`document`
	Any token that identifies a named type from a specific mashable information source. The `service` attribute must be used with named types. With this complex type, mashable metadata provides type information for the token.

Variables that are implicitly declared always have a document type. For more details, see also:

- "Valid Date Formats for MashZone NextGen Mashables and Mashups" on page 179
- "Commonly Supported Output Character Encodings" on page 461

For example:

```
<output name="result" type="document"/>
<input name="securityType" type="svc:certType" service="Credentials"/>
<input name="previousDays" type="boolean"/>
<variables>
  <variable name="loops" type="number"/>
  <variable name="amznResult" type="document"/>
  <variable name="user.email" type="string"/>
  <variable name="today" type="datetime" default="2010-07-01"/>
</variables>
```

Document type variables are represented in XML and must be *well-formed*. The MashZone NextGen Server automatically converts the results of component mashable information sources to XML for any variable that holds complex objects, unless the variable has a subtype specified.

Document-type variables can also contain well-formed HTML or JSON by specifying a `subtype`. These subtypes:

- Allow mashups or macros to properly parse HTML. See "Handling HTML Responses" on page 623 for examples.
- Allow mashups or macros to prevent conversion of JSON responses or input parameters to XML. See "Handling JSON Responses or Inputs" on page 624 for examples.

You can also construct complex data for variables or parameters as literal XML. See "Default and Constructed Values" on page 454 for information and examples on constructing data.

Default and Constructed Values

Most parameters and variables are assigned data by an EMML statement or as parameters passed into the mashup or macro when it is invoked.

You can define default values for variables and parameters with a scalar datatype. Use the `default` attribute. You cannot set a default for variables or parameters with a complex datatype. For example:

```
<input name="queryDate" type="date" default="2007-03-01"/>
...
<variables>
  <variable name="key" type="string" default="1SF8BTKBTXN6XP68BY02"/>
  <variable name="httpResponse.Content-type" type="string"
        default="text/csv; charset=UTF-8"
</variables>
```

You can, however, construct the data for a variable or parameter that has a complex datatype by defining the XML and data literally within <variable>, <input> or <output>.

Note: If you define the content of <output> using literal XML, this is the only output for the mashup or macro.

Add the XML within <variable>, <input> or <output> and enter the literal data. It is a good practice to define a separate namespace for this literal XML to clearly differentiate from EMML. For example:

```
<inputparam name="query">
  <svc:query>
    <svc:category>books</svc:category>
    <svc:ranking>allResults<svc:ranking>
    <svc:date>{$queryDate}</svc:date>
    <svc:maximum>100</svc:maximum>
  </svc:query>
</inputparam>
<variables>
  <variable name="header" type="document">
    <header>
      <Authorization>GoogleLogin auth={$auth}</Authorization>
      <Content-Type>application/atom+xml</Content-Type>
    </header>
  </variable>
  <variable name="soapRequest" type="document">
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:enc="http://schemas.xmlsoap.org/soap/encoding" >
      <soap:Header>{$soapHeader}</soap:Header>
      <soap:Body>{$soapBody}</soap:Body>
    </soap:Envelope>
  </variable>
</variables>
```

You can also construct variables with dynamic data using some EMML statements. See "Constructing the Mashup Result, Inputs or Intermediate Variables" on page 539 for links and information.

Variable and Parameter Scopes

Variables may have a global scope or a local scope. Variables that have a global scope can be used in any statement in the mashup after they have been declared (explicitly or implicitly). Variables with a local scope are only valid within the statement where

they are declared or any descendant statements. They are not valid until they have been declared.

If both global and local variables use the same name, the local variable is used within its scope and the global variable is used everywhere else.

Input and output parameters have a global scope when they are declared in <mashup> or <operation>. Parameters declared in <macro> are only valid within the macro itself. Mashups that use macros, however, do have access to the macro result. See "Calling a Macro" on page 647 for more information.

Variables declared explicitly in <mashup> or <operation> or declared implicitly by any statement that is a direct child of <mashup> or <operation> have a global scope. Variables declared explicitly within EMML statements or declared implicitly by nested statements have a local scope of that containing statement.

With a mashup such as:

```
<mashup name="VariableRules"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
      ../schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML" >
  <output name="result" type="document"/>
  <variables>
    <variable name="explicitGlobal" type="string" default="Crichton"/>
  </variables>
  <assign literal="Ken Scott" outputvariable="implicitGlobal"/>
  <display message="main explicitGlobal: " expr="upper-case($explicitGlobal)"/>
  <display message="main implicitGlobal: " expr="upper-case($implicitGlobal)"/>
  <for variable="i" startCounterValue="1" finalCounterValue="1">
    <!-- explicit local variable, hides global variable within for loop -->
    <variable name="explicitGlobal" type="string" default="Steig Larsson"/>
    <display message="for loop explicitGlobal: "
       expr="upper-case($explicitGlobal)"/>
    <!-- Updates implicitly created global variable -->
    <assign literal="Kathryn Stockett" outputvariable="implicitGlobal"/>
    <display message="for loop implicitGlobal: "
      expr="upper-case($implicitGlobal)"/>
    <assign literal="Ken Scott" outputvariable="implicitLocal"/>
    <display message="for loop implicitLocal: "
      expr="upper-case($implicitLocal)"/>
    <variable name="explicitLocal" type="string" default="Ian M. Banks"/>
    <display message="for loop explicitLocal: "
      expr="upper-case($explicitLocal)"/>
  </for>
  <display message="main explicitGlobal:" expr="upper-case($explicitGlobal)"/>
  <display message="main implicitGlobal:" expr="upper-case($implicitGlobal)"/>
</mashup>
```

The debugging messages showing variable states would be:

```
main explicitGlobal: CRICHTON
main implicitGlobal: KEN SCOTT
for loop explicitGlobal: STEIG LARSSON
for loop implicitGlobal: KATHRYN STOCKETT
for loop implicitLocal: KEN SCOTT
for loop explicitLocal: IAN M. BANKS
main explicitGlobal: CRICHTON
main implicitGlobal: KATHRYN STOCKETT
```

Referencing Parameters and Variables

You refer to parameters or variables in mashup scripts within XPath expressions or in dynamic mashup expressions in the form:

`$`*parameter-or-variable-name*

> **Note:** There is one exception to this rule. For the *inputparameters* property of <invoke>, omit `$` and use just the variable or parameters names.

This is also true for variables that you declare implicitly.

In this first example, an existing parameter or variable named `variableA` is used in an XPath expression to assign a portion of this variable to a new, implicitly declared variable named `implicitVar`.

```
<assign fromexpr="$variableA/item/name" outputvariable="$implicitVar"/>
...
<assign fromvar="$inputParamA" outputvariable="$declaredVariable"/>
```

The second example simply refers to an existing input parameter, `inputParamA`, and an existing variable, `declaredVariable`.

Working Samples

Every sample mashup in MashZone NextGen contains an <output> declaration and most samples use either <variable> or <input> declarations. Some working samples that may be of particular interest include:

- KMLCdataEncoding (cdata.emml) for CDATA encoding variables or output
- i4lnFeeds (encoding.emml) for character encoding
- VariableScoping (scope.emml) for scoping rules on variable declarations
- GoogleWebClipping (webclipping.emml) for handling responses in HTML format

See "Mashup Samples" on page 659 for a complete list of sample mashups and where to find them.

<input>

Optional element to declare parameters that can be used as input to mashup, operation or macro. Input parameters can be used to provide values for EMML statements including input parameters for component mashables.

See also "Declaring Mashup and Macro Variables and Parameters" on page 452 for more information and examples.

Can Contain	Typically empty. The input data can be defined as text and any well-formed literal XML.

Allowed In	mashup | macro | operation

Attributes

Name	Required	Description
name	yes	The required name for a variable, input parameter or output parameter. Parameter names must be unique within the scope of the mashup script. Names for variables must be unique within the scope of the mashup, if the variable has a global scope, or within the statement in which the variable is declared. See also "Valid Names for Variables and Parameters" on page 452 for reserved names and other rules.
type	yes	The required data type for this input parameter. Valid values for this include: ■ `boolean` ■ `date` (*deprecated*, use `datetime` instead) ■ `datetime` ■ `document` (complex, structured data) ■ `number` ■ `string` ■ Any token that identifies a data type defined by a published mashable. Tokens are typically used for input or output parameters where the service has metadata for a named datatype. See also `subtype`.
subtype		For document-type input parameters, optionally defines the content as `JSON`. For JSON inputs, subtype ensures that the JSON input is *not* converted to XML. Currently, input parameters with a JSON `subtype` are supported *only* for the POST body for <directinvoke>. See "Handling JSON Responses or Inputs" on page 624 for more information and examples.

Name	Required	Description
service		The optional name of the published mashable that has metadata defining the data type for this parameter or variable. This is required if the value of the type attribute is a token.
default		The optional, simple default value to use for this variable, input parameter or output parameter. You cannot set default values for document-type variables or parameters.
label		The optional label to display for this input parameter. Typically, this is used in Wires or in apps.

<output>

The parameter that holds the result returned from a mashup, operation or macro.

See also "Declaring Mashup and Macro Variables and Parameters" on page 452 for more information and examples.

Can Contain	Typically empty. Can contain text and any well-formed literal XML. If content is defined within this statement, this is the sole output for the mashup or macro.
Allowed In	mashup | macro | operation

Attributes

Name	Required	Description
name	yes	The required name for this output parameter. Parameter names must be unique within the scope of the mashup or macro.
type	yes	The required data type for the result of this mashup or macro. Valid values for this include: ■ `boolean` ■ `date` (*deprecated*, use `datetime` instead) ■ `datetime`

Name	Required	Description
		■ `document` (complex, structured data) ■ `number` ■ `string` ■ Any token that identifies a data type defined by a published mashable. Tokens are typically used for input or output parameters where the service has metadata for a named datatype.
subtype		Optionally further defines the datatype of document-type results for this mashup or macro: ■ `HTML` ■ `JSON` If this is not present, document-type output is represented as a well-formed XML document. **Note:** Currently, `subtype` is only supported for parameters and variables for <mashup>, <macro> and <directinvoke>. Setting `subtype="HTML"` can help prevent parsing errors for HTML results from web sites or web services. For web services that return JSON responses, setting `subtype` ensures that the JSON response is *not* converted to XML. **Note:** With responses left in JSON format, you *cannot* process the response with other EMML statements except <script> using JavaScript code. See "Handling JSON Responses or Inputs" on page 624 for more information and examples.
service		The optional name of the registered mashable that has metadata defining the data type for this parameter or variable. This is required if the value of the type attribute is a token.
default		The optional, simple default value to use for this variable, input parameter or output parameter.

Name	Required	Description
		You cannot set default values for document-type variables or parameters.
label		The optional label to display for this output parameter. Typically, this is used in Wires or in apps.
output-cdata-section-elements		A list of element names, separate by whitespace, whose text content should be wrapped in CDATA sections in this output. For more information and examples, see "Wrapping Results or Variables in CDATA Sections" on page 619.
output-encoding		The character encoding to use for the output of this mashup. This defaults to UTF-8. This may be any valid character encoding supported by the JDK used by the application server that hosts the MashZone NextGen Server. See "Commonly Supported Output Character Encodings" on page 461 for common valid values.

Commonly Supported Output Character Encodings

The actual character encodings that are supported for mashup output is determined by the JVM in use with the MashZone NextGen Server when the mashup is run. See JDK documentation for full details.

Some of the most common character encodings include:

Encodings		Description
`ASCII`	`US-ASCII`	ASCII or the American Standard Code for Information Interchange
`Big5`		Traditional Chinese
`CP852`	`IBM852`	MS_DOS Latin 2
`CP1250`	`Windows-1250`	Windows Eastern European
`CP1251`	`Windows-1251`	Windows Cyrillic

Encodings		Description
CP1252	Windows-1252	Windows Latin-1
GB2312	EUC_CN	EUC encoding, Simplified Chinese
EUC-JP	EUC_JP	EUC encoding, Japanese
EUC-KR	EUC_KR	EUC encoding, Korean
ISO-8859-1	ISO8859_1	Latin alphabet 1
ISO-8859-2	ISO8859_2	Latin alphabet 2
ISO-8859-5	ISO8859_5	Latin/Cyrillic alphabet
ISO-8859-7	ISO8859_7	Latin/Greek alphabet
ISO-8859-8	ISO8859_8	Latin/Hebrew alphabet
ISO-8859-9	ISO-8859-9	Latin alphabet 5
KOI8-R	KOI8_R	Russian
Shift-JIS	SJIS	Japanese
UTF-8	UTF-8	8-bit, UCS Transformation Format
UTF-16	UTF_16	16-bit, UCS Transformation Format

<variables>

A list of variables to use for input, output or to hold any intermediate data or document content in the flow of mashup processing. Variables must be declared as direct children of <mashup>, <macro>, <operation> or any looping statement that can contain other EMML statements.

Variables declared in <mashup> or <operation> have a global scope and are accessible in any statement. Variables declared in <macro> or looping statements have only a statement scope. Variables must be declared explicitly or implicitly before they are used.

See also "Declaring Mashup and Macro Variables and Parameters" on page 452 for more information and examples.

Can Contain	(variable+)
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

<variable>

One variable to hold input, output or any intermediate data or document content.

Can Contain	Typically empty, but can contain text and any well-formed literal XML.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name	yes	The required name for a variable, input parameter or output parameter. Parameter names must be unique within the scope of the mashup script. Names for variables must be unique within the scope of the mashup, if the variable has a global scope, or within the statement in which the variable is declared.
type	yes	The required data type for this variable. Valid values for this include: ■ `boolean` ■ `date` (*deprecated*, use `datetime` instead) ■ `datetime` ■ `document` (complex, structured data) ■ `number` ■ `string` ■ Any token that identifies a data type defined by a published mashable. Tokens are typically used for input or output parameters where the service has metadata for a named datatype.

Name	Required	Description
		■ `schema`, an extension type for dataset schemas used with the Real-Time Analytics Query Language. See "RAQL Extensions for Dataset Schemas" on page 887 for more information. ■ `variable:`*schema-type-variable-name* , refers to variable of that name within the mashup that has a type of `schema`. This schema-type variable defines datatype and optional path or other information for the dataset in this variable. This is an extension type for variables that hold datasets used with the Real-Time Analytics Query Language. See "RAQL Extensions for Dataset Schemas" on page 887 for more information. See also `subtype`.
subtype		Optionally further defines the datatype of a document-type variable. If `subtype` is not present, document-type variables are represented as well-formed XML. **Note:** Currently, `subtype` is only supported for parameters and variables for the <mashup>, <macro>, <directinvoke> and <raql> statements. Valid subtypes include: ■ `HTML` Setting `subtype="HTML"` can help prevent parsing errors for HTML results received from web sites or web services. ■ `JSON` For web services that return JSON responses, setting `subtype` ensures that the JSON response is *not* converted to XML. **Note:** With responses left in JSON format, you *cannot* process the response with other EMML statements except <script> using JavaScript code. See "Handling JSON Responses or Inputs" on page 624 for more information and examples.

Name	Required	Description
		■ `CSV` This is an extension for datasets used with the Real-Time Analytics Query Language. See "Supported Data Formats for RAQL" on page 771 for more information.
service		The optional name of the published mashable that has metadata defining the data type for this parameter or variable. This is required if the value of the type attribute is a token.
default		The optional, simple default value to use for this variable, input parameter or output parameter. You cannot set default values for document-type variables or parameters.
output-cdata-section-elements		A list of element names, separate by whitespace, whose text content should be wrapped in CDATA sections when output. For more information and examples, see "Wrapping Results or Variables in CDATA Sections" on page 619.
datafile		An extension attribute for use with other extensions from the Real-Time Analytics Query Language. See "RAQL Extensions to <variable> for File Data Sources" on page 886 for more information.
stream		An extension attribute for use with other extensions from the Real-Time Analytics Query Language. See "RAQL Extensions to EMML Statements for Streaming" on page 885 for more information.

Declaring Namespaces

You must declare these namespaces in a mashup script:

- Namespaces used in component mashable information sources.

> **Note:** When you declare namespaces for component mashables, you *must* use the same namespace prefix as the MashZone NextGen Server.

- Namespaces used in the construction of input parameters, output parameters or variables.
- The *macro reference namespace* for any custom statements (macros) that you use in your mashup. See "Calling a Macro" on page 647 for more information.
- The EMML namespace and the MashZone NextGenEMML Extension namespace.

Note: See "EMML Namespaces" on page 450 for the currently supported namespaces for EMML and macros.

You declare namespaces using the *xmlns* attribute on any element in a mashup script. Typically, it is easier to define all the namespaces used in the script on the <mashup> element. For macros, especially in a macro library, it is a good practice to define namespaces on <macro>.

This example shows <mashup> with the default namespaceDe set to EMML:

It has two additional namespace declarations, for Atom and XHTML, that are used in the <constructor> statement to build the body of a <directinvoke> that posts to a blogging service. Namespaces are simply valid URIs - they can be URLs or URNs.

Note: Previous releases used <namespace> elements to define namespace for component mashables. This technique is *deprecated*, although it is still supported.

Declaring Data Sources

If your mashup script invokes SQL queries or statements directly, you typically refer to named datasources that have already been configured in the MashZone NextGen Server called *implicit datasources*. If needed, you can explicitly declare a <datasource> with connection information for a database directly in the mashup script. You can use several named datasources in a mashup script as well as a single, explicitly declared, default datasource.

The advantages, disadvantages and configuration requirements for implicit and explicit datasources include:

Datasource	Advantages	Disadvantages	Configuration
Implicit Configured in the MashZone NextGen Server by MashZone NextGen administrators. **Tip:** It is a best practice to use implicit datasources.	■ Single, consistent configuration. ■ Easily managed in clustered environments. ■ Credentials for database access are secure (encrypted in the MashZone NextGen Repository). ■ Supports 'hot' updates to drivers without restarting the MashZone NextGen Server.	■ Does not support dynamic connection information. Connection and credential information is static in the MashZone NextGen Repository.	
Explicit Declared in the mashup script using the <datasource> element.	■ Can support dynamic connection information or credentials using MashZone	■ Credentials and connection information is less secure, as clear text within the mashup script	You must add and configure drivers in the MashZone NextGen Server for explicit datasources. See

Datasource	Advantages	Disadvantages	Configuration
	NextGen attributes or input parameters to the mashup script.	or defined as input parameters or MashZone NextGen attributes. ■ Configuration is in individual mashup scripts, and thus harder to manage and maintain consistency. ■ If datasources are named, connection information *must* be consistent across all mashup scripts in the MashZone NextGen Repository.	"Configuring Datasource Drivers" on page 472 for instructions.

Using implicit named datasources in EMML is a best practice. See also "Named Datasources versus the Default Datasource" on page 468 for more information.

See "Explicitly Defining the Connection" on page 469 and "Dynamic Connections" on page 470 for instructions on declaring datasources explicitly. See "Working Samples" on page 470 for information on mashup samples for datasources.

Named Datasources versus the Default Datasource

Implicit datasources are named when they are configured in the MashZone NextGen Server. You use this name in SQL statements in mashup scripts to identify which datasource to apply the SQL statements to.

Explicit datasources can also have a name that you can use in SQL statements to identify which datasource to apply the SQL statements to. Named datasources:

- Provide better performance because the MashZone NextGen Server caches named connections.

Because of this caching, datasource names for each connection *must be unique and consistent* across all mashups published in one MashZone NextGen Server.

- Allow you to work with multiple datasources in one mashup script.

If the name is omitted in the <datasource> declaration, the datasource is considered a default datasource. SQL statements without a datasource name are applied to the default datasource.

In this example, the datasource on the localhost is treated as the default datasource. Two additional explicit datasources, named HR and Finance are declared:

```
<datasource url="jdbc:oracledb:osql://localhost:1521"
  driverclassname="oracle.jdbc.driver.OracleDriver"
  username="system" password=""/>
<datasource url="jdbc:oracledb:osql://234.10.25.2:1521"
  driverclassname="oracle.jdbc.driver.OracleDriver"
  name="Finance" username="system" password=""/>
<datasource url="jdbc:oracledb:osql://234.10.35.2:1521"
  driverclassname="oracle.jdbc.driver.OracleDriver"
  name="HR" username="system" password=""/>
```

Explicitly Defining the Connection

Each <datasource> declaration must have one of these two sets of properties:

- A URL that defines the JDBC connection and credential information to log into the database. For example:

  ```
  <datasource url="jdbc:hsqldb:hsql://localhost:9001"
    username="system" password="sa"/>
  ```

  The form of the `url` depends on the type of database. Common URL patterns include:

  - `jdbc:hsqldb:`*driver-type* `://`*hostname* `:`*port*
  - `jdbc:mysql://`*hostname* `/`*databasename*

    Important: For MySQL databases, it is *recommended* that you include the database name in data source URLs. If this information is omitted, you may experience access errors when running or debugging the mashup script.

  - `jdbc:oracle:`*driver-type* `@`*hostname* `:`*port*
  - `jdbc:postgresql://`*hostname* `:`*port* `/`*database-name*
  - `jdbc:sqlserver://`*hostname* `:`*port* `;databaseName=`*database-name*
  - `jdbc:sybase:Tds:`*hostname* `:`*port*

  Both the `username` and `password` attributes are also required, but `password` can be empty if no password is required. If security is a concern for datasource credentials included in mashup scripts, define datasources in the MashZone NextGen Server and use implicit data sources instead.

In most cases, you also *must* include a driver class to define the JDBC driver to use with the connection. This is optional, but if you omit the driver class, the MashZone NextGen Server uses the HSQL driver for the default MashZone NextGen Repository. For example:

```
<datasource url="jdbc:oracledb:osql://localhost:9001"
  driverclassname="oracle.jdbc.driver.OracleDriver"
  username="system" password=""/>
```

- A JNDI name for connections to the database. For example:

```
<datasource jndiname="java:/comp/env/jdbc/myDatasource" />
```

The JNDI name must be in the form `java:/comp/env/jdbc/`*jndi-name*. No other properties for <datasource> are needed with JNDI connections.

Dynamic Connections

Like most EMML attributes, connection information can be set dynamically using MashZone NextGen parameters, input parameters or variables. This example retrieves connection information from MashZone NextGen global attributes:

```
<variables>
  <variable name="global.customerDS.url" type="string"/>
  <variable name="global.customerDS.driver" type="string"/>
  <variable name="global.customerDS.user" type="string"/>
  <variable name="global.customerDS.pw" type="string"/>
</variables>
<datasource name="customerDS"
   url="$global.customerDS.url"
   driverclassname="$global.customerDS.driver"
   username="$global.customerDS.user"
   password="$global.customerDS.pw"/>
```

Working Samples

For working examples of the <datasource> declaration, see the following sample mashups scripts:

- DatabaseSampleJNDI (jndids.emml)
- DatabaseSample (sql.emml)

See "Mashup Samples" on page 659 for a complete list of sample mashups and where to find them.

<datasource>

Optional element to explicitly declare the connection information to a database. Connection configuration can be specified as a JDBC connection or using JNDI. This connection information is used to directly execute SQL commands.

Tip: In most cases, you do not need to declare datasources for SQL Statements in mashups. Instead, identify the datasource using the names assigned to datasources that have already been configured in the MashZone NextGen Server. See "Declaring Data Sources" on page 467 for more information.

You can declare any number of named datasources in a mashup script.EMML statements can also refer to implicit datasources that have been configured in the MashZone NextGen Server and are not declared in the mashup script. You can declare and use a single, unnamed datasource that is considered the default. EMML statements that execute SQL commands and do not have a datasource name use the default datasource.

Datasource names must be *unique* within a given MashZone NextGen Server. Connection information for a given datasource name must be identical as connections for named data sources are cached. Caching is not used for default datasource connections.

See also "Declaring Data Sources" on page 467 for more information and examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		Optional name for this datasource. If omitted, this becomes the default datasource for EMML SQL statements that do not define a datasource.
url		The URL for a JDBC connection to this datasource in the form `jdbc:name:protocol://host:port`. If omitted, you must specify connection information for this datasource in the `jndiname` attribute.
driverClassName		The class name of the driver for a JDBC connection to this datasource. This is only relevant if `url` is specified. If omitted, this defaults to the HSQLDB JDBC driver class.
username		The user name for the JDBC connection to this datasource. This is required for JDBC connections (in `url`), but is not relevant for JNDI connections. **Note:** If security for datasource credentials in mashup scripts is a concern, configure datasources in the MashZone NextGen

Name	Required	Description
		Server instead and use implicit datasources in mashup scripts.
password		The password to log into this datasource with. This is required, but can be empty, for JDBC connections (in `url`). It is not relevant for JNDI connections. **Note:** If security for datasource credentials in mashup scripts is a concern, configure datasources in the MashZone NextGen Server instead and use implicit datasources in mashup scripts.
jndiname		The JDNI name for this datasource, in the form `java:/comp/env/jdbc/`*jndi-name* . If omitted, you must specify JDBC connection information in the `url` and related attributes.

Configuring Datasource Drivers

If your mashup script uses a <datasource> statement with explicit connection information to access a database and execute direct SQL commands, you must add the JAR files containing JDBC drivers for your data source to the MashZone NextGen Server classpath and add configuration for these drivers.

Note: You do *not* need to add JDBC driver JARs for datasources used in mashups when the mashup script uses implicit datasources. Implicit datasources refer to a named datasource that has already been configured in the MashZone NextGen Server.

1. Copy the JAR file for your database driver to the folder for JAR files in *MashZoneNG-config* .

   This is either an external configuration folder for the MashZone NextGen Server or the default location *web-apps-home* /presto/WEB-INF/lib.

   Important: It is a best practice is use an external configuration folder, especially for clustered environments.

2. Restart the MashZone NextGen Server.

Invoking Component Mashups and Mashables

Information sources for your mashup can be other mashups or specific operations for mashable information sources. To invoke a mashable operation or another mashup and include the results in mashup processing, you add either:

- <invoke> for MashZone NextGen mashups or mashables. See "<invoke>" on page 473, "Automatically Generating <invoke> Statements in Mashup Editor" on page 681 and "Controlling Component Mashup/Mashable Invocation at Runtime" on page 491 for details.
- <directInvoke> for ungoverned, publically accessible web services or web sites. See "<directinvoke>" on page 478 and "Controlling Component Mashup/Mashable Invocation at Runtime" on page 491 for details.

  Note: Ungoverned web services must be either REST, SOAP or syndicated web feeds.

  See also "Web Clipping with a Mashup Script" on page 621 for additional techniques.

<invoke>

This statement invokes an operation for a registered MashZone NextGen mashable information source or another mashup.

For more information and examples, see:

"Example 6. <invoke> Basics" on page 475

"Example 7. Passing Input Parameters" on page 475

"Example 8. Handling Mashable Logins" on page 476

"Example 9. Adding SOAP Headers" on page 477

"Example 10. Filtering Mashable Results Immediately" on page 477

"Example 11. Syntax for Results with Arrays of Complex Objects" on page 477

See also "Working Samples" on page 478 for sample mashups in MashZone NextGen for <invoke>.

You can quickly and easily create this statement in Mashup Editor. See "Automatically Generating <invoke> Statements in Mashup Editor" on page 681.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
service	yes	The mashable or mashup ID for the publishedMashZone NextGenmashable to invoke.
serviceversion		Optional. Reserved for future use.
operation	yes	The operation ID to invoke in this mashable or mashup.
inputvariables		An optional list of input or variable names, separated by commas, to use as input parameters to the operation. Variables must be listed in the order in which the mashable or mashup operation expects input parameters. **Important:** Do not use the `$` when referring to parameters or variables.
outputvariable		The name of the variable to hold the response from this invocation. This is optional.

header		The name of a variable containing headers to use when this mashable or mashup is invoked. The payload in a headers variable must be in the form:
		`<header>`
		`  service-specific payload for headers`
		`</header>`

responseheader		The name of the variable to receive headers in the response from this invocation. This is optional.
timeout		The maximum number of seconds to wait for a response before terminating this invocation. The default timeout is 5 minutes.
onerror		The behavior for the mashup script if errors occur when this mashable or mashup is invoked. Valid values include: - `abort` = Stops all further mashup script processing after an invocation error. If <invoke> is inside a "<try>" on page 563 block, any

Name	Required	Description
		exceptions may still be caught and handled by <catch> blocks. ■ `continue` = Continues mashup script processing after an invocation error. **Note:** If <invoke> is in a <try> block with `onerror` set to `continue`, the mashup *ignores* any <catch> blocks. See "Lightweight Error Handling for Component Mashups and Mashables" on page 574 for more information.
filterexpr		An optional XPath expression to use as a filter for the result of this mashable or mashup.

<invoke> Examples

<invoke> Basics

You must identify the mashable or mashup and operation name that you want to invoke. In most cases, you should also identify an output variable to receive the results from the invocation.

Note: You can fill in mashable or mashup and operation information automatically in the Mashup Editor. See "Automatically Generating <invoke> Statements in Mashup Editor" on page 681 for more information.

For example:

```
<invoke service="ArtimaDev" operation="getFeed"
    outputvariable="$artima"/>
```

In most cases, the results from the mashable or mashup are complex documents. For mashables or mashups that return simple data, it is a best practice to explicitly declare the variable for the result and the datatype. For example:

```
<variable name="myName" type="string"/>
<invoke service="LookUpSvc" operation="getName"
    outputvariable="$myName"/>
```

Passing Input Parameters

Specify input parameters for the mashable operation in the `inputvariables` attribute. If there are multiple input parameters, list the names of the <variable> or <inputparam> elements to provide this data in the *expected order*, separated by commas.

Important: Do *not* use the `$` prefix for parameter or variable names in this attribute.

Also, when you invoke another mashup, input parameters must be listed in the order in which they appear in the invoked mashup script.

For example:

```
<invoke service="AmazonDocStyle" operation="ItemLookup"
  inputvariables="lookup" outputvariable="$reviews"/>
<invoke service="AmazonREST" operation="ListSearch"
  inputvariables="aws, date1, key, op, listType, email"
  outputvariable="$result"/>
```

If you omit parameters in `inputvariables`, the effect depends on the placement of the parameter:

- Parameters that you omit at the end of the list are not sent. The receiving mashable information source or mashup may use a default value, if any, handle the omission or treat this as an error.
- If you skip a parameter in the list in `inputvariables`, the parameter is sent as a null value. For mashups, this overwrites any default value for the parameter. Mashables may handle this or may treat this as an error.

Handling Mashable Logins

For REST web services, WSDL web services or Syndicated feeds that require basic HTTP authentication or NTLM authentication, you can also send user credentials to automatically handle login for the mashable invocation. If you omit credentials in <invoke>, MashZone NextGen uses the default credentials, if any, that were provided when the mashable was published.

Note: For mashables that use SSL and x.509 certificates for authentication, MashZone NextGen automatically handles authentication with the mashable based on SSL configuration and WSS policies, if relevant.

You must construct a header with the appropriate credentials and then identify it using the `header` attribute in <invoke>. Basic HTTP authentication looks something like this:

```
<constructor outputvariable="basicHttp">
  <header>
    <httpBasicAuth>
      <username>someone@xyz.com</username>
      <password>somepassword</password>
    </httpBasicAuth>
  </header>
</constructor>
<invoke service="someRESTService" operation="someOp" header="$basicHttp"
   outputvariable="$result" />
```

For NTLM credentials, the structure of header that you use in <invoke> is slightly different:

```
<constructor outputvariable="ntlm">
  <header>
    <ntlmAuth>
      <username>someone@xyz.com</username>
      <password>somepassword</password>
      <domain>somedomain</domain>
    </ntlmAuth>
```

```
  </header>
</constructor>
<invoke service="someRESTService" operation="someOp" header="$ntlm"
   outputvariable="$result" />
```

Because <constructor> allows "Dynamic Mashup Expressions" on page 604, you can also set user credentials dynamically. This example passes in user credentials as input to the mashup:

```
<input name="thisUser" type="string"/>
<input name="thisPW" type="string"/>
...
<constructor outputvariable="basicHttp">
  <header>
    <httpBasicAuth>
      <username>{ $thisUser }</username>
      <password>{ $thisPW }</password>
    </httpBasicAuth>
  </header>
</constructor>
<invoke service="someRESTService" operation="someOp" header="$basicHttp"
   outputvariable="$result" />
```

Adding SOAP Headers

For WSDL web services that require headers, construct the header in <constructor> and use the `header` attribute in <invoke>.

```
<constructor outputvariable="myheader">
  <header>
    <serviceHeader>
      <ser:Header>
        <ser:Username>user@xyz.com</ser:Username>
      </ser:Header>
    </serviceHeader>
  </header>
</constructor>
<invoke service="someWSDLService" operation="someOp" inputvariable="lookup"
  header="$myheader" outputvariable="$result" />
```

Filtering Mashable Results Immediately

You can also optionally filter the result of the mashable operation using the `filterexpr` attribute and an XPath 2.0 expression. This is a shortcut to specifying a separate <filter> statement. For example:

```
<invoke service="ArtimaDeveloper" operation="GET" outputvariable="$artima"
  filterexpr="/rss/channel/item[contains(description, 'Scala')]"/>
```

Syntax for Results with Arrays of Complex Objects

When the results from the mashable information source contain an array of complex objects, the results contain a list of XML nodes for each complex object in the array. This list is enclosed within a parent node with a name in the form *object-name* `_Array`.

For a list of Customer objects, for example, the result would look something like this:

```
...
<Customer_Array>
  <Customer>
     <!-- customer detail structure here -->
```

```
  </Customer>
  <Customer>
     <!-- customer detail structure here -->
  </Customer>
  ...
</Customer_Array>
...
```

Note: In previous releases, results for arrays of complex objects used a different naming convention for the parent node of the array:

```
<object-name-Array> ... </object-name-Array>
```

You must update XPath expressions in mashup scripts from earlier releases to use the new name, *object-name* `_Array`, instead.

Working Samples

See "Mashup Samples" on page 659 for a complete list of mashup samples and where to find them.

<directinvoke>

Use <directinvoke> to invoke web services or web sites that are publicly accessible on the Web and thus not governed by MashZone NexGen. Web services that you invoke directly must have a REST, SOAP or syndicated feed (RSS/Atom) interface.

Note: For a sample using <directinvoke> with a SOAP server, see the soapservice.emml sample mashup.

You can use a proxy server when mashups invoke services with <directinvoke>. This is determined by proxy server configuration for the MashZone NextGen Server.

You can also disable the use of <directinvoke> in mashups to prevent anyone from invoking services that are not governed by MashZone NextGen.

For examples and more information, see the following:

"Example 12. <directinvoke> Basics for GET Requests" on page 484

"Example 13. Passing Parameters in <directinvoke>" on page 485

"Example 14. <directinvoke> Basics for POST Requests" on page 485

"Example 15. <directinvoke> Basics for PUT and DELETE Requests" on page 487

"Example 18. Getting Headers, Status Codes and Cookies from <directinvoke> Responses" on page 489

"Example 20. Dynamic Endpoints or Parameters for <directinvoke>" on page 490

"Example 21. <directinvoke> with a Security Profile" on page 490

"Handling HTML Responses" on page 623

"Example 16. Adding HTTP Headers to <directinvoke> Requests" on page 488

"Example 17. Adding HTTP Basic Authentication to <directinvoke> Requests" on page 488

"Handling JSON Responses or Inputs" on page 624

"Working Samples" on page 491

Can Contain	securityprofile?
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
endpoint	yes	The URI to the endpoint for the publicly accessible web service or web site to invoke. This can be an explicit URL or dynamic using variables. See "Example 20. Dynamic Endpoints or Parameters for <directinvoke>" on page 490 for an example. The endpoint may use HTTPS and SSL for secure connections. <directinvoke> supports one-way SSL connections *only*. Certificates for secure endpoints are validated against certificates in the *trust store* configured for the JRE (Java Runtime Engine) where the MashZone NextGen Server is running when the mashup script is run. **Note:** The default JRE Trust Store may or may not be the same Trust Store that is configured for the MashZone NextGen Server, which can cause errors. You must update Java configuration to avoid this problem. For certificates issued by well-known certificate authorities, no additional configuration is required. For self-signed certificates or certificates from other certificate authorities, however, you may simply add the public certificate for the endpoint to the default trust store to allow successful invocations.

Name	Required	Description
method		The HTTP method to use invoking this web site or web service: ■ `GET` ■ `POST` ■ `PUT` ■ `DELETE` Other attributes for <directinvoke> depend on the method used. For more information, see: ■ "Example 12. <directinvoke> Basics for GET Requests" on page 484 ■ "Example 14. <directinvoke> Basics for POST Requests" on page 485 ■ "Example 15. <directinvoke> Basics for PUT and DELETE Requests" on page 487
requestbody		An optional attribute for the name of the variable containing the body of the request to send when `method` is `POST`, `PUT` or `DELETE` and the content is *not* name/value pairs. The payload of the request body may be: ■ Well-formed XML. The variable containing this payload must be a document-type. ■ Well-formed JSON. The variable containing this payload must be a document-type. ■ A string in any form except name/value pairs. The variable containing this payload must have a string datatype. See "Example 14. <directinvoke> Basics for POST Requests" on page 485 and "Handling JSON Responses or Inputs" on page 624 for more information.

header		The name of a document-type variable containing HTTP or web-service headers to use when this web or web site is invoked. The payload in a headers variable must be in the form:
		`<header>`
		`  web-service-specific header payload`
		`  -- or --`
		`  <http-header-name>value</http-header-name>`
		`</header>`

Name	Required	Description
		See "Example 16. Adding HTTP Headers to <directinvoke> Requests" on page 488 for an example defining an HTTP header. See also "Example 17. Adding HTTP Basic Authentication to <directinvoke> Requests" on page 488.
outputvariable	yes	The required variable to accept the output of this statement. For responses with complex data, this variable must have a document datatype. The response data can be either: ■ Well-formed XML ■ HTML ■ JSON With HTML or JSON responses, the subtype specified for the variable to contain the response affects how the response is parsed and converted. See "Handling HTML Responses" on page 623 and "Handling JSON Responses or Inputs" on page 624 for more information and examples.
feedtype		An optional attribute for syndicated feeds identifying the protocol for normalization. See "Normalization and MashZone NextGen Support for RSS/Atom Formats" on page 132 for more information. Valid values include: ■ `native` = does not normalize feed results. ■ `rss` = normalizes the feed results to RSS 2.0. ■ `atom` = normalizes the feed results to Atom 1.0.
followredirects		An optional attribute to explicitly control whether <directinvoke> follows redirect directives from the endpoint: ■ If omitted, redirects are accepted when `method` is `GET` and are ignored for any other method type. ■ `true` = follow endpoint redirects regardless of the method. ■ `false` = ignore endpoint redirects regardless of the method.
htmlparser		The HTML parser to use to fix an HTML response if the response is not well-formed: ■ `tagSoup` = the default parser for versions 3.8 and later.

Name	Required	Description
		■ `jtidy` = JTidy was the parser used in mashups for version 3.7 or earlier. Choose this option if the mashup needs backwards compatibility.
responseheader		An optional attribute for the name of the variable to receive HTTP headers from the response.
responsecode		An optional attribute for the name of the variable to receive the HTTP status code from the response.
cookies		An optional attribute for the name of the variable to receive cookies from the response.
bypassproxy		An optional attribute to have the invocation bypass the proxy server, if any. This is `false` by default.
stream		An extension attribute for use with other extensions from the Real-Time Analytics Query Language. See "RAQL Extensions to EMML Statements for Streaming" on page 885 for more information.
timeout		The maximum number of seconds to wait for a response before terminating the invocation to this web service or web site. The default timeout is 5 minutes.
onerror		The behavior for the mashup or macro if errors occur when this web service or web site is invoked. Valid values include: ■ `abort` = Stops all further mashup or macro processing after an invocation error. If <directinvoke> is located inside a "<try>" on page 563 block, any exceptions may still be caught and handled by subsequent <catch> blocks. ■ `continue` = Continues mashup or macro processing after an invocation error. **Note:** If <directinvoke> is in a <try> block with `onerror` set to `continue`, the mashup or macro *ignores* any <catch> blocks. See "Lightweight Error Handling for Component Mashups and Mashables" on page 574 for more information.

securityprofile

The security profile to use to support secure connections with this web site or web service. For more information on security profiles, see "Mashable Authentication with Security Profiles" on page 161.

Important: Currently, <directinvoke> supports *only* the `CAS2` security profile bundled with MashZone NextGen.

A security profile is not required for secure connections to web sites or web services using one-way SSL. <directinvoke> does not support mutual SSL.

See "Example 21. <directinvoke> with a Security Profile" on page 490 for examples.

Can Contain	param*
Allowed In	directinvoke

Attributes

Name	Required	Description
id	yes	The name of the security profile to use for this endpoint. **Note:** Currently, you can *only* use the `CAS2` security profile. In most cases, no additional information is needed for this profile. If, however, the URL where the CAS Server should validate tokens for this web site or web service is *not* the endpoint URL for the web site or web service, you must add one <param> element to provide the verification URL. See "Example 21. <directinvoke> with a Security Profile" on page 490 for examples.

param

One security property to use with this security profile. Typically, security properties include credentials or other information used to authenticate the user for this request or enforce secure connections with this web site or web service.

The exact properties you need to supply depend on the security profile that you are using. See "Mashable Authentication with Security Profiles" on page 161 for information on the properties to use for the security profiles that are bundled in MashZone NextGen.

For information on properties for custom security profiles, contact your MashZone NextGen administrator.

See "Example 21. <directinvoke> with a Security Profile" on page 490 for examples.

Can Contain	Empty
Allowed In	securityprofile

Attributes

Name	Required	Description
name	yes	The name of the security property to pass in this security profile to support secure communications with this endpoint.
value	yes	The value for this security property.

directinvoke Examples

<directinvoke> Basics for GET Requests

Add a <directinvoke> element to the mashup script. Identify the `endpoint` (the URL) to the web service or web site, the HTTP `method` (GET or POST) to use to send the request and the `outputvariable` to receive the results from the web service or site.

Note: URLs sometimes include characters, such as `&` which *must* be escaped in XML. See "XML Escaping in URLs and Expressions" on page 600 for more information.

You can also set `bypassproxy` to bypass the proxy server for this request.

If the endpoint is for a syndicated feed, you can also identify the format you want results normalized to in the `feedtype` attribute. Choose `atom` = ATOM 1.0, `rss` = RSS 2.0, or `native` = do not normalize the results.

Note: For Google News feeds, there is a known issue when `feedtype = "atom"`. Simply remove the `feedtype` attribute to avoid this problem.

For example:

```
<directinvoke endpoint="http://www.myCompany.com/rest-services/getNames"
    method="GET" outputvariable="$result"/>
<directinvoke endpoint="http://www.AnotherSite.com/getInfo"
    method="GET" bypassproxy="true" outputvariable="$result"/>
<directinvoke endpoint="http://rss.news.yahoo.com/rss/topstories"
    method="GET" feedtype="rss" outputvariable="$news"/>
```

Passing Parameters in <directinvoke>

You can pass parameters to the web service or web site using any attribute name that is *not* defined in EMML. These attributes can also have a namespace, if needed.

Note: In most cases, parameters are used with the `GET` method. Some web services, however, require the `POST` method with name/value pairs that are *url-encoded* and have only simple values. In this case, you add these as named parameters. For more information, see "Example 14. <directinvoke> Basics for POST Requests" on page 485

In the following example, the attributes `query` and `appID` are not defined in EMML. These will be passed as parameters to the web site.

```
<directinvoke endpoint="http://www.myCompany.com/products/getItems"
    method="GET" outputvariable="$result" query="items=all"
    appID="67GYH30N25" />
<directinvoke endpoint="http://www.svcsltd.com/getReservation"
    method="POST" outputvariable="$news" xmlns:sc="http://www.svcltd.com/"
    sc:date="20070515" sc:nights="3"/>
```

The second example shows the use of parameters with the `POST` method and includes a a namespace with additional attributes for request parameters. This namespace must be defined on the <mashup> element or on <directinvoke> itself.

<directinvoke> Basics for POST Requests

Web services may require complex parameters or input in a specific format, such as Atom, JSON, SOAP or other, vendor-specific formats. In these cases, you use the `POST` method in <directinvoke>. Web services may also simply specify that requests must use the `POST` method.

With the `POST` method, the parameters or other information for the request is usually specified in the body of the request. Because the format may be different, the request *must* also set the HTTP Content-Type header (sometimes called the MIME-type).

There are two basic patterns to use <directinvoke> with the `POST` method, based on the format of the content:

- *Name/Value Pairs*: in this case, the web service expects named parameters with *simple* values. The HTTP content-type is `application/x-www-form-urlencoded`.

  For this type of content, you simply set the `method` attribute to `POST` and add the parameters as attributes just as you do for `GET` requests. The MashZone NextGen Server automatically adds the HTTP Content-Type header with a value of `application/x-www-form-urlencoded`. See "Example 13. Passing Parameters in <directinvoke>" on page 485 for an example.

- *Any Content That Is Not Name/Value Pairs*: this includes any content-type other than `application/x-www-form-urlencoded` or content that has complex parameters. Common examples are SOAP content, any XML content such as Atom + XML, XHTML, HTML, JSON or vendor-specific content.

  For this type of content, you:

- Define the information for the body of the request in a <constructor> or <variable> statement or any statement where you can build a variable with literal XML content. See "Constructing the Mashup Result, Inputs or Intermediate Variables" on page 539 for more information and examples.

  The mashup can also receive the body as an input parameter. For example, you can send the content for a JSON request body. See "Handling JSON Responses or Inputs" on page 624 for more information and examples.

- Use the `requestbody` attribute to point to the variable with the request content.
- If the web service requires name/value pairs as well as a complex body in the request, you *cannot* use both the `requestbody` attribute and request-specific attributes for the named parameters as you would for `GET` requests. Instead, set named parameters in the `endpoint` attribute and set parameter values explicitly or with dynamic mashup expressions. For example:

  ```
  <directinvoke outputvariable="$blogresult" method="post"
     endpoint="http://postURI?namedParam={$value1}"
     requestbody="$xmlPost" header="$header"/>
  ```

  See "Dynamic Mashup Expressions" on page 604 for more information.

- Explicitly set the appropriate value for the HTTP Content-Type header in a variable. See "Example 16. Adding HTTP Headers to <directinvoke> Requests" on page 488 for an example.
- Use the `header` attribute to point to the HTTP headers to send in the request. See "Example 16. Adding HTTP Headers to <directinvoke> Requests" on page 488 and "Example 17. Adding HTTP Basic Authentication to <directinvoke> Requests" on page 488 for more information and examples.

This example shows a request body in Atom format:

```
<constructor outputvariable="xmlPost">
  <entry xmlns='http://www.w3.org/2005/Atom'>
    <title type='text'>My First Blog</title>
    <content type='xhtml'>
      <div xmlns="http://www.w3.org/1999/xhtml">
        <p>This is my first blog and I'm not sure what to say.</p>
        <p>"Hello World!" just seems silly.</p>
      </div>
    </content>
    <author>
      <name>Mia</name>
      <email>mia@xyz.com</email>
    </author>
  </entry>
</constructor>
<constructor outputvariable="header">
  <header>
    <Content-Type>application/atom+xml</Content-Type>
  </header>
</constructor>
<directinvoke endpoint="$bloggerURL" outputvariable="$blogresult"
    method="post" requestbody="$xmlPost" header="$header"/>
```

<directinvoke> Basics for PUT and DELETE Requests

The requirements for direct invocations with the `PUT` or `DELETE` methods are somewhat similar to those for `POST`:

- `PUT` invocations typically have a request body that provides the data for the endpoint to add or update. `DELETE` may also have a request body, although this is less common.

  Use the `requestbody` attribute to point to the variable with the content for the request body.

  If the web service requires name/value pairs as well as a complex body in the request, you *cannot* use both the `requestbody` attribute and request-specific attributes for the named parameters as you would for `GET` requests. Instead, set the named parameters and values in the `endpoint` attribute. See "Example 14. <directinvoke> Basics for POST Requests" on page 485 for an example.

- The format and content for the request body depends *entirely* on what the endpoint requires. It can be XML, JSON or any endpoint-specific string format.

  - For XML content, define the information for the body of the request in a <constructor> or <variable> statement or any statement where you can build a variable with literal XML content. See "Constructing the Mashup Result, Inputs or Intermediate Variables" on page 539 for more information and examples.

  - For JSON or any vendor-specific text format, build the content using <assign> or <script>. You can also pass the JSON as an input parameter to the mashup.

    See "Handling JSON Responses or Inputs" on page 624 for information and examples.

- Depending on endpoint requirements, you may also need to add specific HTTP headers to clarify the purpose of the request or the data type of the request body.

  Use the `header` attribute to point to the HTTP headers to send in the request. See "Example 16. Adding HTTP Headers to <directinvoke> Requests" on page 488 for more information and examples.

This example shows the EMML for a PUT request to an endpoint for a CouchDB database. This endpoint requires a JSON object that is either an insert or an update to an existing 'document' in the database, based on the `mode` input parameter:

```
..
<input name="docId" type="string"/>
<input name="mode" type="string"/>
<input name="member" type="string"/>
<variables>
  <variable name="idProp" type="string"/>
  <variable name="nameProp" type="string"/>
  <variable name="statusProp" type="string"/>
  <variable name="reqBody" type="string"/>
</variables>
<!-- build JSON object for request body -->
<assign fromexpr="concat('{"id":"',$docId,'",')" outputvariable="$idProp"/>
<assign fromexpr="concat('{"name":"',$member,'",')"
  outputvariable="$nameProp"/>
```

```
<assign fromexpr="concat('{"status":"',$mode,'"}')"
  outputvariable="$statusProp"/>
<assign fromexpr="concat($idProp,$nameProp,$statusProp)
  outputvariable="$reqBody"/>
<if condition="$mode='new'">
  <directinvoke endpoint="http://myorg.com/memberDB/updates"
    method="put" outputvariable="$updateResult" requestbody="$reqBody"/>
  <else>
    <header>
      <If-Match>{$docId}</If-Match>
    </header>
  <directinvoke endpoint="http://myorg.com/memberDB/updates"
    method="put" outputvariable="$updateResult" requestbody="$reqBody"
    header="$header" />
  </else>
</if>
...
```

For inserts with this example, no HTTP header is required. For updates, however, the endpoint uses the HTTP If-Match header to identify which 'document' to update in the database.

Adding HTTP Headers to <directinvoke> Requests

You can define HTTP headers in variables and pass them in the request to the web service or web site using the `header` attribute. The variable for the header must be a document type with a <headers> root node. Each HTTP header is an XML element child and the value for that header is the content of the XML element.

Note: You can use this syntax to pass basic HTTP authentication headers to a web service or web site.

For example:

```
<operation name="directWithHeaders">
  <variable name="httpHeader" type="document">
    <headers>
      <Content-type>application/x-www-form-urlencoded</Content-type>
    </headers>
  </variable>
...
  <directinvoke endpoint="http://www.myCompany.com/rest-services/getItems"
    method="GET" outputvariable="$result" header="$httpHeader" />
...
```

Adding HTTP Basic Authentication to <directinvoke> Requests

Another common requirement is to send basic authentication information to the web service using the HTTP Authorization header. This requires that the username and password assigned to this header be encoded using Base64 as a single string.

You can use the built-in macro, `computeBasicAuth` to perform the Base64 encoding. This macro has two input parameters, for the user name and password, and returns the encoded string. For example:

```
  <input name="username" type="string" default="user"/>
  <input name="password" type="string" default="pw"/>
  <variable name="basicauth" type="string"/>
  <macro:computeBasicAuth user="$username" password="$password"
```

```
    outputvariable="$basicauth"/>
  <variable name="httpHeader" type="document">
    <headers>
      <Authorization>{$basicauth}</Authorization>
    </headers>
  </variable>
...
  <directinvoke endpoint="http://www.myCompany.com/rest-services/getItems"
    method="GET" outputvariable="$result" header="$httpHeader" />
...
```

Getting Headers, Status Codes and Cookies from <directinvoke> Responses

The body of the response from the web service or web site is placed in the output variable you specify. You can use the following attributes in <directinvoke> to define variables to hold header and other information from the response:

- `responseheader` = a variable to hold the HTTP headers from the response.
- `responsecode` = a variable to hold the HTTP status code from the response.
- `cookies` = a variable to hold any cookies returned by the response.

This example checks for the HTTP redirect status and invokes the web service if it is detected:

```
<operation name="directWithCookiesEtc">
  <variables>
    <variable name="result" type="document" />
    <variable name="redirecturl" type="string" />
    <variable name="responseHeader" type="document" />
    <variable name="responseCode" type="string" />
    <variable name="cookies" type="document" />
  </variables>
  <directinvoke endpoint="http://myCompany.com/some-service/getDoc"
    method="POST" outputvariable="$result"
    responseheader="$responseHeader" responsecode="$responseCode"
    cookies="$cookies" />
  <display message="Headers are" variable="$responseHeader" />
  <display message="Cookies are" variable="$cookies" />
  <display message="ResponseCode is" variable="$responseCode" />
  <if condition="number($responseCode)=302">
    <assign fromexpr="$responseHeader//Location/string()"
      outputvariable="$redirecturl" />
    <display message="Redirect to " variable="$redirecturl" />
    <directinvoke endpoint="$redirecturl" method="GET"
      responseheader="$responseHeader" outputvariable="$result"
      responsecode="$responseCode" cookies="$cookies" />
  </if>
...
```

Handling Redirects from the Endpoint

By default, <directinvoke> will follow redirects specified by the endpoint when the `method` is GET, but *ignores* endpoint redirects for other HTTP methods. You can explicitly control how <directinvoke> handles endpoint redirects using the `followredirects` attribute.

The examples shown below override the default behavior to ignore redirects for a GET method and follow redirects for a POST method:

```
<directinvoke endpoint="http://www.myCompany.com/products/getItems"
    method="GET" outputvariable="$result" followredirects="false"/>
<directinvoke endpoint="http://www.svcsltd.com/getReservation"
    method="POST" outputvariable="$news" followredirects="true" />
```

Dynamic Endpoints or Parameters for <directinvoke>

You can use the <template> declaration to allow the endpoint or parameters for <directinvoke> to be set dynamically. This technique uses dynamic mashup expressions to resolve the URL. For example:

```
...
  <!-- allow users to select a symbol -->
  <input name="ticker" type="string" default="GOOG"/>
  <!-- variable used to construct the dynamic endpoint -->
  <variables>
    <variable name="result" type="document" />
    <variable name="wholeURL" type="string" />
  </variables>
  <!-- template to construct dynamic endpoint -->
  <template expr="http://finance.yahoo.com/q/pr?s={$ticker}"
    outputvariable="$wholeURL"/>
  <directinvoke endpoint="$wholeURL" method="POST" outputvariable="$result" />
...
```

See "Dynamic Mashup Syntax" on page 651 for more information and example on using <template>.

<directinvoke> with a Security Profile

For endpoints that require a security profile to pass authentication or other secured connection information, you use <securityprofile> as a child of <directinvoke> to supply this information and <param> children in <securityprofile>.

MashZone NextGen includes security profiles for some very common security requirements that you may use. See "Mashable Authentication with Security Profiles" on page 161 for information on these bundled profiles.

MashZone NextGen administrators can also add custom security profiles to MashZone NextGen. Contact your MashZone NextGen administrator for information on custom security profiles, if any.

Important: Currently, <directinvoke> supports *only* the `CAS2` security profile bundled with MashZone NextGen.

A security profile is not required for secure connections to web sites or web services using one-way SSL. <directinvoke> does not support mutual SSL.

This example shows a direct invocation using the MashZone NextGen built-in CAS2 security profile:

```
...
  <directinvoke endpoint="http://myorg.com/securedservice"
    method="POST" requestbody="$body" outputvariable="$result" >
    <securityprofile id="CAS2"/>
  </directinvoke>
...
```

The security profile, in this case, does not require any additional information. In rare cases, you may need to supply a verification URL for the secured endpoint where the CAS Server can verify security tokens, such as in this example:

```
...
  <directinvoke endpoint="http://myorg.com/securedservice"
    method="POST" requestbody="$body" outputvariable="$result" >
    <securityprofile id="CAS2">
      <param name="casServiceUrl"
        value="http://myorg.com/securedservice/verifyTokens"/>
    </securityprofile>
  </directinvoke>
...
```

Working Samples

Many of the sample mashup scripts for MashZone NextGen use <directinvoke>. Some samples of particular interest include:

- YahooHotJobsFilter (rssfilter.emml) for the basics on using GET

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Controlling Component Mashup/Mashable Invocation at Runtime

There are two attributes you can use in either <invoke> or <directinvoke> to define runtime invocation behavior for component mashups or mashable information sources. Both attributes are optional:

- Use the *timeout* attribute to set a maximum number of seconds to wait for a response. The default timeout is 5 minutes (300 seconds).
- Use the *onerror* attribute to determine mashup or macro behavior when invocation errors occur *outside* a <try> block.

  Note: If you use this attribute when the <directinvoke> or <invoke> statement is contained in a <try> statement, it overrides all <catch> statements.

  The value for `onerror` can be either `continue` to have the mashup or macro continue processing or `abort` to stop any further processing. Abort is the default.

  If you use the `continue` option, the MashZone NextGen Server considers the exception handled and returns error information. See "Lightweight Error Handling for Component Mashups and Mashables" on page 574 for more information.

The ErrorHandlingSample (onerror.emml) sample mashup for a working example of both of these attributes. See "Mashup Samples" on page 659 for a complete list of samples and where to find them.

Issuing SQL Statements Directly to a Datasource

In addition to invoking Database mashables, you can issue individual SQL statements on datasources in a mashup script. You can send queries with the <sql> statement or

issue any other SQL statement with <sqlUpdate>. EMML also includes statements to manage SQL transactions (see "SQL Transactions" on page 560). You can also invoke stored procedures directly with <sqlcall>.

Note: Although EMML supports issuing SQL statements on multiple databases within one mashup script, it does *not* support distributed transactions.

MashZone NextGen administrators must define data sources for the databases that you want to work with in the Admin Console.

In rare cases, you may also need to explicitly define datasources in mashup scripts. See "Declaring Data Sources" on page 467 and "Configuring Datasource Drivers" on page 472 for more information.

<sql>

Use <sql> to issue individual SQL queries to a named datasource that has been configured by a MashZone NextGen administrator in the MashZone NextGen Server or a datasource that you have explicitly declared in the mashup script.

Note: In previous releases, the <sql> and <sqlUpdate> statements were also used to invoke stored procedures. This usage is still supported but *deprecated*.

You must first define configuration information and name datasources in the MashZone NextGen Server. This ensures that drivers are available in the classpath. It also keeps database credentials securely encrypted and simplifies EMML code.

You can disable the <sql> command to prevent anyone from using direct SQL queries in a mashup.

For more information and examples, see:

- "Example 22. <sql> Basics" on page 494
- "Handling Table or Column Names with Spaces" on page 494
- "Example 23. Explicitly Declaring the Datasource" on page 495
- "Example 24. Passing Query Parameters" on page 496
- "Example 25. Valid SQL Syntax" on page 496
- "Guarding Against SQL Injection Attacks" on page 496
- "Example 26. Defining a Subset of Rows to Return" on page 496
- "Example 27. Optimizing Performance for Large Datasets" on page 497
- "Example 28. Result Sets" on page 497
- "Example 29. Using Templates to Query Tables Dynamically" on page 498

See also "Working Samples" on page 498 for information on working samples you can review.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		The name of the datasource for this query. In most cases, this is the name of a datasource that has been configured in the MashZone NextGen Server, although the datasource can be declared directly in the mashup script. If omitted, the query is sent to the default datasource.
query	yes	The query to execute. The actual SQL syntax supported depends on the database and driver class for this datasource.
startrow		Optionally, the index for the first row in the result set to return. If omitted, this defaults to one.
rowcount		Optionally, the total number of rows in the result set to return. If omitted, this defaults to the end of the result set.
fetchSize		Optionally, the amount of memory, expressed as an optimal number of rows, that the mashup should request from the datasource at one time. If the total number of rows requested by this SQL statement is larger than this number, the mashup uses multiple requests to the datasource to obtain the full result set to return. This is most useful with SQL statements that may work with a large dataset in the database to optimize performance. This attribute can be used with or without the `rowcount` attribute.
stream		An extension attribute for use with other extensions from the Real-Time Analytics Query Language. See "RAQL Extensions to EMML

Name	Required	Description
		Statements for Streaming" on page 885 for more information.
outputvariable	yes	The required variable to accept the output of this statement.

<sql> Examples>

<sql> Basics

Queries put the SELECT statement in a `query` attribute and set an `outputvariable` to receive the result set. Identify the datasource by name, matching a datasource configured in the MashZone NextGen Server.

For example:

```
...
<variables>
  <variable name="serviceId" type="string" default="CustomerDAO"/>
  <variable name="cnt" type="number"/>
  <variable name="customerId" type="number" default="1001"/>
  <variable name="customers" type="document"/>
</variables>
...
<sql name="auditDS"
  query="select count(*) as countAll from AUDITABLE_EVENT where
  SERVICE_ID = :serviceId" outputvariable="$cnt"/>
<sql name="customerDS"
  query="select CUST_ID, CUST_NAME from CUSTOMERS
  where CUST_ID = :customerId" outputvariable="$customers" />
```

The output variable is typically a document type, to receive a result set, but can be any type.

Handling Table or Column Names with Spaces

For some databases, table, veiw of column names can include spaces. This can cause errors in queries or other SQL statements in EMML code.

The solution is to enclose these names in quote marks. Since the SQL statements are contained as attribute values in EMML, however, simply typing the quote marks to add them causes syntax errors as quote marks and apostrophes are both XML delimiters.

To avoid this, you must use the escaped characters (the XML entities) to enclose names with spaces. See "XML Escaping in URLs and Expressions" on page 600 for a complete list of entities for XML delimeters.

For example:

```
...
<sql name="auditDS"
  query="select count(*) as countAll from &quot;Audit Table&quot;
  where &quot;Service ID&quot; = :serviceId"
  outputvariable="$cnt"/>
...
```

During parsing, the `"` entities are replaced with quote mark (") characters to ensure the SQL runs successfully.

Explicitly Declaring the Datasource

You use the `name` attribute to identify a named datasource for the query. In almost all cases, it is a best practice to configure datasources in the MashZone NextGen Server and simply refer to them by name in EMML code. This simplifies configuration and ensures that database credentials are securely encrypted.

If rare cases, it is useful to explicitly declare the datasource in the mashup script, although this is less secure. Although optional, you should also provide names for explicit datasources as this allows the MashZone NextGen Server to cache connections and enhance performance.

If you omit `name` in <sql>, the query is invoked against the *default datasource* identified in an unnamed <datasource> declaration in the mashup script.

Tip: Datasource names for a given connection *must* be unique across all mashup scripts hosted in a MashZone NextGen Server.

The query shown below would be invoked against the default datasource for the mashup script:

```
<!-- default data source -->
<datasource url="jdbc:hsqldb:hsql://localhost:9001"
  username="system" password="sa"/>
...
<sql query="select count(*) as countAll from AUDITABLE_EVENT where
 SERVICE_ID = :serviceId" outputvariable="$cnt"/>
```

This second query, however, would be invoked against the datasource named `customerDS`.

```
<!-- customer data source -->
<datasource url="jdbc:hsqldb:hsql://234.20.1.65:9001"
  username="system" password="sa" name="customerDS"/>
...
<sql query="select name from category order by name desc"
  name="customerDS" outputvariable="$customers" />
```

And this last query.would be invoked against the datasource named `inventoryDS` whose connection information is provided dynamically using MashZone NextGen global attributes.

```
<variables>
  <variable name="global.inventory.ds.url" type="string"/>
  <variable name="global.inventory.ds.user" type="string"/>
  <variable name="global.inventory.ds.pw" type="string"/>
</variables>
...
<!-- inventory data source -->
<datasource url="$global.inventory.ds.url"
  username="$global.inventory.ds.user"
  password="$global.inventory.ds.pw" name="inventoryDS"/>
...
<sql query="select item from inventory order by PART_NO"
  name="inventoryDS" outputvariable="$items"/>
```

Passing Query Parameters

To use parameters in the SQL query, you must declare <variable>s and use the variable name in the SQL query in the form `:`*variable-name* . For example:

```
...
  <variable name=serviceId" type="string" default="CustomerDAO"/>
...
<sql query="select count(*) as countAll from AUDITABLE_EVENT where
 SERVICE_ID = :serviceId" outputvariable="$cnt" name="auditDS"/>
```

Important: For information on security concerns for query parameters, see "Guarding Against SQL Injection Attacks" on page 496.

Valid SQL Syntax

EMML has no specific requirements about the syntax of the SQL you use in queries. What is valid depends on the datasource the mashup connects to and the driver class used for the connection.

Guarding Against SQL Injection Attacks

Using input parameters in a mashup script to provide query parameters or to build dynamic queries can potentially allow SQL injection attacks in *some* cases. EMML uses Java Prepared Statements to execute queries in <sql> statements. The form of the dynamic portions of the statement determine whether a SQL injection attack is possible.

Queries that use the `:variable-name` syntax to supply *only* the value of a condition in a WHERE clause are not vulnerable to SQL injection because the variable is not interpreted as SQL code. Queries built as shown in "Example 24. Passing Query Parameters" on page 496 have no risk of an injection attack.

There is some risk of SQL injection, however, in dynamic queries built from <template> or <assign> statements where some portion of the resulting query comes from an input parameter. The example shown in "Example 29. Using Templates to Query Tables Dynamically" on page 498:

```
<input name="table" type="string"/>
<variable name="query" type="string"/>
...
<!-- build dynamic query SQL -->
<template expr="select * from {$table}" outputvariable="$query"/>
<!-- invoke dynamic query -->
<sql name="myDataSource" query="$query" outputvariable="$result"/>
```

Clearly can include SQL code other than a simple table name in the input parameter and thus could be open to a SQL injection attack.

Defining a Subset of Rows to Return

You can also define the specific rows you want returned from the query using the `startrow` and `rowcount` attributes. Both of these attributes are optional.

To see the third through the twelfth result, for example, you would do something like this:

```
<sql query="select name from category order by name desc"
  startrow="3" rowcount="10" name="customerDS"
  outputvariable="$customers" />
```

You can use variables with these attributes and omit either `startrow` or `rowcount`. For example:

```
<variables>
  <variable name="hundred" type="number" default="100"/>
  <variable name="hundredone" type="number" default="101"/>
</variables>
...
<!-- row 101 or greater -->
<sql query="select name from category order by name desc"
  startrow="$hundredone" name="customerDS"
  outputvariable="$customers" />
...
<!-- first 100 rows -->
<sql query="select name from category order by name desc"
  rowcount="$hundred" name="customerDS"
  outputvariable="$customers" />
```

Optimizing Performance for Large Datasets

If the SQL query is working with a large dataset or potentially returning a large result set, you can use the `fetchSize` attribute to optimize mashup performance and prevent out of memory issues. This attribute defines a maximum number of rows to fetch from the database in one chunk.

With this example:

```
<sql query="select name from category order by name desc"
  startrow="10000" rowcount="5000" fetchSize="2000"
  name="customerDS" outputvariable="$customers" />
```

The mashup must actually iterate through the first 15,000 rows to build the result set with rows 10,000 through 14,999. A request to obtain all 15,000 rows might fail, for lack of memory, or take an unacceptable amount of time.

Instead, the result set is built with several requests to the database, each returning 2,000 rows. Any rows not needed in the result set for the mashup are simply discarded.

Result Sets

The structure of the result set for a direct SQL query has this form:

```
<records>
  <record>
    <columnA-name>value</columnA-name>
    <columnB-name>value</columnB-name>
    ...
  </record>
</records>
```

Column names are the SQL names from the database.

Using Templates to Query Tables Dynamically

You can also use "<template>" on page 653 declarations to query tables dynamically. The <template> declaration allows you to build the SQL query from input variables and then use that within a <sql> command. For example:

```
<input name="table" type="string"/>
<output name="result" type="document"/>
<variables>
  <variable name="query" type="string"/>
</variables>
...
<!-- build dynamic query SQL -->
<template expr="select * from {$table}" outputvariable="$query"/>
<!-- invoke dynamic query -->
<sql name="myDataSource" query="$query" outputvariable="$result"/>
```

Important: For information on security for dynamic queries, see "Guarding Against SQL Injection Attacks" on page 496.

Working Samples

Two of the mashup samples for MashZone NextGen use the <sql> statement:

- DatabaseSampleJNDI (jndids.emml)
- DatabaseSample (sql.emml)

See "Mashup Samples" on page 659 for a complete list of samples and where to find them.

<sqlUpdate>

Use <sqlUpdate> to execute any SQL statement that is not a query against a named data source that has been configured by a MashZone NextGen administrator in the MashZone NextGen Server or a datasource that you have declared in the mashup script. This includes basic statements to update database records.

Note: In previous releases, the <sql> and <sqlUpdate> statements were also used to invoke stored procedures. This usage is still supported but *deprecated*.

You must first define configuration information and name datasources in the MashZone NextGen Server. This ensures that drivers are available in the classpath. It also keeps database credentials securely encrypted and simplifies EMML code.

You can disable the <sqlUpdate> command to prevent anyone from using direct SQL statements in a mashup.

For more information and examples, see "Example 30. Inserting, Updating or Deleting Records" on page 499, "Handling Table or Column Names with Spaces" on page 500 and "Example 31. Explicitly Declaring the Datasource" on page 500.

Can Contain	Empty

Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		The name of the data source for this statement. In most cases, this is the name of a datasource that has been configured in the MashZone NextGen Server, although the datasource can be declared directly in the mashup script. If omitted, the query is sent to the default data source.
statement	yes	The SQL statement to execute. The actual SQL syntax supported depends on the database and driver class for this data source.
outputvariable		An optional variable to accept the output of this SQL command.

<sqlUpdate> Examples

Inserting, Updating or Deleting Records

To invoke SQL statements that update database records, define the datasource. Specify the SQL code in a `statement` attribute and define an `outputvariable` to receive the result.

For example:

```
<sqlUpdate name="myDataSource"
    statement="insert into
    credentials(id, user_id, password, cred_type)
    values('1005', :user, :password, :type)"
    outputvariable="$insertResult"/>
...
  <sqlUpdate name="myDataSource"
    statement="delete from credentials where id = :id"
    outputvariable="$delResult"/>
...
  <sqlUpdate name="myDataSource"
    statement="update credentials set
    user_id = 'newuser' where id = '1004'"
    outputvariable="$updateResult"/>
```

If the datasource does not return a result, you can omit the `outputvariable` attribute.

Handling Table or Column Names with Spaces

For some databases, table, veiw of column names can include spaces. This can cause errors in queries or other SQL statements in EMML code.

The solution is to enclose these names in quote marks. Since the SQL statements are contained as attribute values in EMML, however, simply typing the quote marks to add them causes syntax errors as quote marks and apostrophes are both XML delimiters.

To avoid this, you must use the escaped characters (the XML entities) to enclose names with spaces. See "XML Escaping in URLs and Expressions" on page 600 for a complete list of entities for XML delimeters.

For example:

```
...
<sql name="auditDS"
  query="select count(*) as countAll from &quot;Audit Table&quot;
  where &quot;Service ID&quot; = :serviceId"
  outputvariable="$cnt"/>
...
```

During parsing, the `"` entities are replaced with quote mark (") characters to ensure the SQL runs successfully.

Explicitly Declaring the Datasource

You use the `name` attribute to identify a named datasource for the SQL statement. In almost all cases, it is a best practice to configure datasources in the MashZone NextGen Server and simply refer to them by name in EMML code. This simplifies configuration and ensures that database credentials are securely encrypted.

If rare cases, it is useful to explicitly declare the datasource in the mashup script, although this is less secure. Although optional, you should also provide names for explicit datasources as this allows the MashZone NextGen Server to cache connections and enhance performance.

If you omit `name` in <sql>, the SQL statement is invoked against the *default datasource* identified in an unnamed <datasource> declaration in the mashup script.

Tip: Datasource names for a given connection *must* be unique across all mashup scripts hosted in a MashZone NextGen Server.

These two insertions are invoked against different databases:

```
<!-- customer data source -->
<datasource url="jdbc:hsqldb:hsql://234.20.1.65:9001"
  username="system" password="sa" name="customerDS"/>
<!-- inventory data source -->
<datasource url="jdbc:hsqldb:hsql://234.20.2.35:9001"
  username="system" password="sa" name="inventoryDS"/>
...
<sqlUpdate statement="insert into
  customers(cust_id, name, contact, phone)
  values('1005', 'TRG, Inc.', 'Mark Walberg', '4152345678')"
  name="customerDS" outputvariable="$custInsert"/>
...
```

```
<sqlUpdate statement="insert into
  contacts(cust_id, firstName, lastName, title, email)
  values('1005', 'Mark', 'Walberg', 'Purchasing Manager',
    'walberg@trg.com')"
  name="contactsDS" outputvariable="$contactInsert"/>
```

You can also get transaction handling using the SQL transaction commands (see "SQL Transactions" on page 560), but mashups do not support distributed transactions.

<sqlcall>

Use <sqlcall> to invoke stored procedures in a named datasource that has been configured by a MashZone NextGen administrator in the MashZone NextGen Server or a datasource that you have explicitly declared in the mashup script. Stored procedures may return a result set or not.

Note: In previous releases, the <sql> and <sqlUpdate> statements were used to invoke stored procedures. This usage is still supported but *deprecated*.

You must first define configuration information and name datasources in the MashZone NextGen Server. This ensures that drivers are available in the classpath. It also keeps database credentials securely encrypted and simplifies EMML code.

You can disable the <sqlcall> statement to prevent anyone from using direct SQL queries in a mashup.

See "Example 32. Examples" on page 502.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		The name of the datasource for this stored procedure. In most cases, this is the name of a datasource that has been configured in the MashZone NextGen Server, although the datasource can be declared directly in the mashup script. If omitted, the stored procedure is invoked in the default datasource.
query	yes	The stored procedure to execute.
params		A comma-separated list of variables to send as input parameters, if any, to the stored procedure.

Name	Required	Description
		The name(s) of these variables must match the input parameter names in the stored procedure.
types		A comma-separated list of database datatypes, one for each input parameter in the `params` atttribute. List the datatypes in the same order that corresponds to the input parameters. See "Example 32. Examples" on page 502.
hasreturn		Either `true` or `false` indicating whether this stored procedure returns a result set or not.
outputvariable	yes	The required variable to accept the output of this statement.

Examples

The following example executes a stored procedure that has one input parameter on the `customerdb` datasource that has been configured by a MashZone NextGen administrator in the MashZone NextGen Server. This procedure has a result set:

```
...
<!-- input parameter for stored procedure -->
<variable name="custId" type="string" />
<!-- variable for result set -->
<variable name="invoices" type="document" />
<!-- invoke stored procedure against named datasource -->
<sqlcall name="customerdb" query="sp_customer_invoices" params="customId"
    types="VARCHAR" hasreturn="true" outputvariable="$invoices" />
...
```

The next example shows a stored procedure that inserts a new record in the same data source as the previous example. This stored procedure has no result set.

```
...
<!-- input parameters for stored procedure -->
<variable name="custId" type="string" />
<variable name="customerName" type="string" />
<variable name="invoices" type="string" />
<!-- invoke stored procedure -->
<sqlcall name="customerdb" query="sp_customer_insert"
    params="custId, customerName, invoices" type="VARCHAR, VARCHAR, VARCHAR"
    hasreturn="false" outputvariable="$result" />
...
```

Transforming Intermediate Results

There are several mashup statements that you can use in mashup scripts to select specific results, sort results, or otherwise transform the data in variables:

- <assign> to assign values or copy part of all of a variable

- <filter> to filter a variable
- <group> to group data in a variable and optionally filter or sort
- <sort> to sort a variable
- <annotate> to add nodes and data to a variable
- <script> to transform data using user-defined scripting code
- <xslt> to transform data using XSLT stylesheets

Using XPath Functions to Transform Data

Many mashup transformations use XPath expressions to select specific nodes for a statement. These expressions can also be used to apply XPath 2.0 functions as part of the transformation.

Functions can transform data, change datatypes, perform calculations or determine boolean conditions. XPath includes many common functions for working with strings, numbers and dates. You can also use XPath constructor functions to cast data to other datatypes.

For some basic examples of XPath functions that transform data, see "A Brief Introduction to XPath 2.0" on page 596. For detailed information about any of these functions or other functions you can use, see the XPath 2.0 Functions specification or the XPath 2.0 Data Model Types. You can also create custom XPath functions to transform data to meet your specific requirements. See "Defining Custom XPath Functions" on page 601 for more information.

<assign>

Use <assign> to assign values to a variable. Values may be:

- Literal values
- Fragments of another variable
- A whole variable

You use XPath expressions to define the fragments or variables to assign or otherwise modify the resulting variable content.

Consider these general rules when using <assign>:

- You cannot use <assign> to change the content of a variable that contains a MashZone NextGen attribute. Variables for MashZone NextGen attributes are read-only. For more information on MashZone NextGen attributes, see "Using MashZone NextGen Attributes in Mashups" on page 614.
- When working with complex variables that have namespaces, you must include either the exact namespace or `*:` as a namespace wildcard at each step in XPath expressions.

- You can use <assign> to cast strings of well-formed XML to a document-type variable or vice versa. See "Example 35. Copy All of a Variable" on page 505 for an example.
- Assignment between XML and JSON variable types will trigger implicit XML / JSON data transformations.

For examples, see "Example 33. Assign Literal Values" on page 505, "Example 34. Copy Fragments of a Variable" on page 505, "Example 35. Copy All of a Variable" on page 505 and "Example 36. Replace Existing Values" on page 506. See also "Working Samples" on page 506 for a list of sample mashups you can review.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
fromvariable		The variable to copy. <assign> must have either a `fromvariable`, a `fromexpr` or a `literal` attribute.
fromexpr		An expression identifying a variable or variable fragment and any functions to apply before copying the result. <assign> must have either a `fromvariable`, a `fromexpr` or a `literal` attribute.
literal		A literal value to assign. <assign> must have either a `fromvariable`, a `fromexpr` or a `literal` attribute.
toexpr		An XPath expression defining a specific node in a variable that is the target to be assigned. <assign> must have either a `toexpr` or an `outputvariable` attribute.
mode		An optional flag to indicate whether the assignment should `replace` the output node or variable. The most common use is to apply an XPath function and assign the result back to the input node or variable.

Name	Required	Description
outputvariable		The variable that is the target to be assigned. <assign> must have either a `toexpr` or an `outputvariable` attribute.

<assign> Examples

Assign Literal Values

You can assign a literal to a variable, identified by name, or to a specific node within a variable, identified by an XPath expression. For example:

```
<!-- assign literal to a variable -->
<assign literal="Business" outputvariable="$storyType"/>
<!-- assign literal to a node -->
<assign literal="today" toexpr="$someVariable/someNode/date"/>
```

Copy Fragments of a Variable

The syntax to copy fragments from one variable uses an XPath expression to identify the specific nodes to copy. For example:

```
<variables>
  <variable name="queryResult" type="document"/>
  <variable name="itemNames" type="document"/>
  <variable name="rotorItems" type="document"/>
</variables>
...
<assign fromexpr="$queryResult/items/item/name"
  outputvariable="$itemNames"/>
...
<assign fromexpr="$queryResult/items/item[contains(name,'rotor')]"
  outputvariable="$rotorItems"/>
```

The first example selects all item name nodes. The second example uses the predicate `[contains(name,'rotor')]` to select only item nodes whose name contains 'rotor' somewhere in the value.

The previous examples copied fragments to a variable. Fragments can also be copied to a specific node within a variable using the `toexpr` attribute and an XPath expression. For example:

```
<assign fromexpr="$queryResult/items/item[1]/name"
  toexpr="$results/first/name"/>
```

Copy All of a Variable

You can also use <assign> to copy an entire variable to another variable or to a specific node in a variable. Use the `fromvariable` attribute to identify the variable to copy.

One common use case for this is to cast a string of *well-formed XML* to a document-type. For example:

```
<variables>
  <variable name="queryResult" type="string"/>
  <variable name="result" type="document"/>
```

```
</variables>
...
<assign fromvariable="$queryResult" outputvariable="$result"/>
```

You can also cast documents to well-formed XML strings.

Replace Existing Values

Another common usage is to use <assign> to apply an XPath function and replace existing data within a variable. For example, to change the case of one or more nodes.

```
<assign fromexpr="$result/orders/order/shipCode/upper-case()"
toexpr="$result/orders/order/shipCode"/>
```

Working Samples

Many sample mashups for MashZone NextGen use the <assign> statement. Some examples of particular interest include:

- AssignExpressionsSample (assignments.emml) for basic <assign> examples
- DateTimeOperations (datetime.emml) for a sample using an XPath function
- GoogleFinanceNews (googlefinancenews.emml) for <assign> examples illustrating many different XPath expressions
- MacrosSample (macros.emml) and example using a dynamic mashup expression

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<filter>

This statement filters the content of a variable based on a condition and places the result in a new variable. You can also simply update the input variable with filtered results.

This returns a document with only those nodes in the input variable that match the condition. If no nodes match, the result is an empty document, typically just the root node.

The filter condition is defined in an XPath expression. You can use comparisons, position, XPath functions and dynamic expressions in the condition. You can also combine multiple conditions.

Note: You can also filter directly from the <invoke> statement.

For examples, see "Example 37. <filter> Basics" on page 507, "Example 38. Filtering Variables with Namespaces" on page 507, "Example 39. Filtering Comparisons Using Variables or MashZone NextGen Attributes" on page 508, "Example 40. Filtering Based on Position" on page 508, "Example 41. Using XPath Functions or Other Comparison Operations in Filtering" on page 508, "Example 42. Combining Filter Criteria" on page 509 or "Example 43. Using Dynamic Filtering Expressions" on page 509. See also "Working Samples" on page 510 for information on sample mashups using this statement.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
inputvariable	yes	The name of the input variable to filter.
filterexpr	yes	An expression that defines the filter to apply to the input variable. All matching nodes are included in the result.
outputvariable	yes	The required variable to accept the output of this statement.

<filter> Examples

<filter> Basics

Identify the variable to filter in `inputvariable` and the variable to receive the filtered output in `outputvariable`. Define the filter as an XPath 2.0 expression in `filterexpr`. This is a relative expression within the input variable.

For example:

```
<variables>
  <variable name="queryResult" type="document"/>
  <variable name="westCoastOnly" type="document"/>
</variables>
...
<filter inputvariable="$queryResult"
  filterexpr="/customers[region='West']"
  outputvariable="$westCoastOnly"/>
```

The XPath predicate defines the comparison criteria for the filter. See "A Brief Introduction to XPath 2.0" on page 596 for basic information about XPath predicates.

Filtering Variables with Namespaces

The filter expression uses XPath predicates to compare some value in the input variable to a literal value, as shown above. The path in the filter expression to find the input variable node to compare must include any namespaces used, as shown in this example:

```
<!-- select all lists that have user rating info -->
<filter inputvariable="$result" filterexpr="//ns:list[ns:rating]"
  outputvariable="$rate"/>
```

You can also use wildcards to match any namespace as shown in this example:

```
<!-- select all lists that have user rating info -->
<filter inputvariable="$result" filterexpr="//*:list[*:rating]"
  outputvariable="$rate"/>
```

Filtering Comparisons Using Variables orMashZone NextGen Attributes

You can also filter based on any variable defined in the mashup or any MashZone NextGen attribute. See "Using MashZone NextGen Attributes in Mashups" on page 614 for more information on MashZone NextGen attributes and their use in mashups.

You reference variables or MashZone NextGen attributes using *$variable-name* . For example:

```
<input name="selectedEmail" type="string"/>
...
<filter inputvariable="$staff" outputvariable="$result"
  filterexpr="/personnel/person[email = $selectedEmail]"/>
<variables>
  <variable name="global.secureClearance" type="string"/>
</variables>
...
<filter inputvariable="$staff" outputvariable="secure"
  filterexpr="/personally/person[clearance = $global.secureClearance]"/>
```

Filtering Based on Position

When results contain repeating items, you can filter based on the cardinal order or position of items. You can select one specific item by number or select a contiguous group using the XPath function position(). For example:

```
<!-- select the third staff member -->
<filter inputvariable="$staff" outputvariable="$result"
  filterexpr="/personnel/person[3]"/>
<!-- select the first 3 staff members -->
<filter inputvariable="$staff" outputvariable="$result"
  filterexpr="/personnel/person[position() = (1 to 3)]"/>
```

Note: XPath uses 1-based position indexing.

Using XPath Functions or Other Comparison Operations in Filtering

In addition to the = operator, you can use any valid XPath operator in the filtering expression. This includes >, >=, <, <= and != (not equals). For example:

```
<!-- select all lists with user ratings > 5 -->
<filter inputvariable="$result" outputvariable="$highRated"
  filterexpr="//*:list[*:rating &gt; 5]" />
```

Note: You must use XML escaping with XPath operators that use the < or > characters, as this example shows.

You can also use two XPath functions, contains(*string-to-search* ,*string-to-match*) and matches(*string-to-search* ,*pattern-to-match* ,[*flags*]), to handle substring comparisons. Both of these functions look for a string anywhere in the node you specify.

The contains() function does case sensitive comparisons. The example shown below selects all `record` nodes from the `employees` variable whose `fname` child contains "So" exactly. This would match names such as Sophy, Sonja or OnSo, but would not match Jefferson:

```
<!-- find employee with first name containing "So" exactly -->
<filter inputvariable="$employees" outputvariable="$result"
  filterexpr='/records/record[contains(fname,"So")]' />
```

The matches() function can do either case-sensitive or case-insensitive searches. To make the match case-insensitive, pass `i` in the *flags* parameter. You can also use regular expressions in the second parameter to do wildcard matching or handle specific patterns.

The example shown below selects all `record` nodes from the `employees` variable whose `fname` child contains "so" in any case. This would match names such as Sophy, Sonja and Jefferson

```
<!-- find employee with first name containing "so" any case -->
<filter inputvariable="$employees" outputvariable="$result"
  filterexpr='/records/record[matches(fname,"so","i")]' />
```

In addition to these two built-in functions, you can use any other built-in XPath function in a filter expression. XPath contains a wide variety of string, number and date functions to allow you to work with data. See XPath 2.0 functions for more information.

You can also create you own XPath functions and add them to the MashZone NextGen Server to perform specific data transformations to handle specific comparisons that are unique to your organization. See "Defining Custom XPath Functions" on page 601 for more information.

Combining Filter Criteria

You can combine criteria in a filter expression to handle more complicated filtering. Use the keywords `and` or `or` in the expression. This example filters a variable based on any one of several criteria:

```
<filter inputvariable="$employees" outputvariable="$result"
  filterExpr="records/record[matches(fname,$query,"i") or
    matches(address,$query,"i") or
    matches(phone,$query,"i") or
    matches(city,$query,"i") or
    matches(country,$query,"i") or
    matches(dept,$query,"i")]"/>
```

Using Dynamic Filtering Expressions

Filtering expressions contain both a path to a node and a predicate that defines a comparison to use as the filter. You can make the value of that comparison dynamic using variables or MashZone NextGen parameters (see "Example 39. Filtering Comparisons Using Variables or MashZone NextGen Attributes" on page 508).

You can also make the path portion of filtering expressions dynamic using input parameters and *dynamic mashup expressions*. Dynamic mashup expressions enclose references to variables in braces ({ }) to ensure that they are evaluated before the XPath expression itself is evaluated.

This example invokes a MashZone NextGen mashable information source and filters the results based on a path and field identified in input parameters:

```
<!-- input with an XPath path -->
<input name="selectedItem" type="string"/>
<input name="comparisonField" type="string"/>
<!-- invoke with dynamic filter defined by input -->
<invoke service="YahooHotJobsRss" operation="getFeed"
  filterexpr="{$selectedItem}[matches({$comparisonField},'Ruby')]"
  outputvariable="$result"/>
```

See "Dynamic Mashup Expressions" on page 604 for more examples and information.

Working Samples

There are several sample mashups for MashZone NextGen that use the <filter> statement. Some of particular interest include:

- FiltersSample (filters.emml) for basic examples
- DynamicHotJobFilters (dynamicfilter.emml) or DynamicDataFiltering (dynamicfilter2.emml) for examples of dynamic filtering
- MergeUniqueItems (unique.emml) for a sample of using <filter> to find unique values

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<group>

This statement groups repeating nodes by the unique values of a field and optionally performs calculations for each group. You can also filter which repeating nodes are included in each group, if desired. Groups can be nested, allowing any level of detail.

<group> constructs a document with items for each unique group value using literal XML that you define and dynamic expressions to determine content or perform calculations. See "Literal XML Structures with Literal or Dynamic Data" on page 539 for more information.

With this statement, you:

1. Example 44. Define What to Group
2. And construct the result:
   - Example 45. Copy and Sort to Construct the Result
   - Example 46. Include Unique Values in the Result
   - Example 47. Add Calculations to the Result or construct the result solely from calculations

You can also optionally "Example 48. Filter With a Condition" on page 515 to exclude some items from a group or Example 49. Nest Groups to get multiple levels of sorting

and calculations. See also "Working Samples" on page 517 for information on sample mashups that use this statement.

Can Contain	(group | literal XML and dynamic expressions that define the resulting output for the group) Literal XML must be well-formed, starting with a root node. Elements in the structure should use a namespace, so that they are clearly separated from EMML markup.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
by	yes	An XPath expression that defines the set of nodes to be grouped and the field whose unique values define each group. For nested groups, this XPath expression is relative to the XPath expression for the parent group.
having		An XPath expression that defines a criteria to filter the nodes to be included in the group.
outputvariable	yes	The required variable to accept the output of this statement.

<group> Examples

Define What to Group

You use an XPath expression in the `by` attribute to select a set of repeating nodes from a variable and identify the field whose unique values define each group, called the *group key*. You also set the `outpuvariable` attribute to hold the result.

```
<group by="$catalog//book/genre" outputvariable="$groupResult">
</group>
```

This example finds all unique `genre` values (the group key) for `book` nodes from the `$catalog` variable. Keep these points in mind:

- The XPath expression in the `by` attribute does *not* explicitly identify which nodes are the repeating items for the group, called the *group items*. Starting backwards from the final node in this XPath expression, the MashZone NextGen Server uses the first ancestor that repeats as the group items.

- The final node in this XPath is the group key and it must result in a simple value that can be compared. This can be an attribute or any child or descendant that contains simple data.

  By default, this value is treated as a string. You can use functions in the XPath expression to cast key values to other types or to perform case conversions.

- Do *not* use predicates in the `by` attribute to filter which items should be selected. Instead, specify a filtering condition in the `having` attribute.

Copy and Sort to Construct the Result

You construct the grouped result to put into the output variable as literal XML with either literal data or "Dynamic Mashup Expressions" on page 604 to generate data.

Important: The XPath expressions within literal XML for a <group> are *relative* to the repeating node in the `by` attribute.

Inside <group>, add the XML that should be generated as the result of the <group> statement for each group of repeating items. The XML must be well formed. So:

- Each group must be wrapped in a single node.
- You can also create the root node that wraps the entire document for the result.

  Note: If you omit a document root node, the MashZone NextGen Server adds an <output> root node by default, using any namespace you have defined for the group items.

To include all data for each repeating item in the top-level group, use the `{$group_id}` dynamic mashup expression. See "Example 49. Nest Groups" on page 515 for examples of dynamic mashup expressions for subgroups.

This example sorts books in the `$catalog` variable by genre including full copies of each book with the genre group:

```
<group by="$catalog//book/genre" outputvariable="$groupResult">
  <res:bookGenres>
    <genre>{$group_id}</genre>
  <res:bookGenres>
</group>
```

The result for this <group> statement would looks something like this:

```
<res:bookGenres>
  <genre>
    <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <keywords>XML software language nonfiction</keywords>
      <price>44.95</price>
      <copiessold>100</copiessold>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications with XML.</description>
    </book>
    <book id="bk110">
```

```
      <author>O'Brien, Tim</author>
      ...
    </book>
  </genre>
  <genre>
    <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      ...
    </book>
    ....
  </genre>
  ...
</res:bookGenres>
```

Include Unique Values in the Result

You can also pull out the unique values for the group key to include them in the result using the `{$group_key}` dynamic mashup expression. This example sorts books by genre in the `$catalog` variable, adding the name of each genre to the node wrapping the group:

```
<group by="$catalog//book/genre" outputvariable="$groupResult">
  <res:bookGenres>
    <genre name="{$group_key}">{$group_id}</genre>
  <res:bookGenres>
</group>
```

The results would look something like this:

```
<res:bookGenres>
  <genre name="Computer">
    <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <keywords>XML software language nonfiction</keywords>
      <price>44.95</price>
      <copiessold>100</copiessold>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications with XML.</description>
    </book>
    <book id="bk110">
      <author>O'Brien, Tim</author>
      ...
    </book>
  </genre>
  <genre name="Fantasy">
    <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      ...
    </book>
    ....
  </genre>
  ...
</res:bookGenres>
```

Add Calculations to the Result

You can also perform calculations on the repeating items in each group using dynamic mashup expressions and XPath functions. Common examples include generating a count of items in the group or summing a numeric field for all items in the group. You can:

- Add these calculations to a group containing the sorted items.

  You can use any XPath function that can perform calculations based on a set of nodes (a sequence). The path for this set of nodes *must* be a relative path to the repeating items for this group.

  This example calculates the number of books in a given genre and adds this to the sorted output:

```
<group by="$catalog//book/genre" outputvariable="$groupResult">
  <books>
    <genre name="{$group_key}" count="{count(title)}">
    {$group_id}</genre>
  </books>
</group>
```

  The `count` function accepts a set of nodes as input. In this case the relative path points to the `title` node in each book (the repeating item). The result for this <group> statement would look something like this:

```
<books>
  <genre name="Computer" count="4">
    <book id="bk101"> ... </book>
    ...
  </genre>
  <genre name="Fantasy" count="5">
    <book id="bk102"> ... </book>
    ...
  </genre>
  ...
</books>
```

- Or omit repeating item data and construct the result *solely* from group keys and calculations. This is useful if you only want statistics, not the actual data.

  This example outputs just the name of book genres from the `$catalog` variable and calculates the total number of copies sold for that genre:

```
<group by="$catalog//book/genre" outputvariable="$groupResult">
  <genre name="{$group_key}" copiessold="{sum(copiessold)}"/>
</group>
```

  Note: As with all XPath expressions inside the constructed result for <group>, the dynamic mashup expression `{sum(copiessold)}` uses a path that is relative to the repeating item for <group>.

  The result for the <group> statement would look something like this:

```
<output>
  <genre name="Computer" sold="140"/>
  <genre name="Fantasy" sold="1600"/>
  <genre name="Horror" sold="1000"/>
  <genre name="Romance" sold="700"/>
```

```
  <genre name="Science Fiction" sold="400"/>
</output>
```

Filter With a Condition

You can optionally filter group items using a condition in the `having` attribute. This next example selects unique genres for all books whose prices are greater than 5:

```
<group by="$catalog//book/genre" having="price &gt; 5"
    outputvariable="$groupResult">
   ...
</group>
```

The condition in `having` is an XPath expression that is *relative to the group items* defined in the `by` attribute. This filter affects both the repeating items included in the result and any calculations for the group.

Nest Groups

You can nest <group> elements within the literal XML of the result to get additional levels of grouping. The initial <group> statement defines the repeating items to be group. Nested <group> statements simply define additional levels of grouping.

The following example groups books in the `$catalog` variable by genre and author. It includes all book data:

```
<group outputvariable="$groupedResult" by="$catalog//book/genre">
<books>
  <genre name="{$group_key}" copiessold="{sum(copiessold)}">
  <group by="author">
    <author name="{$group_key}" count="{count(title)}">
      {$group_id1}
    </author>
   </group>
  </genre>
</books>
</group>
```

The results would look something like this:

```
<res:bookGenres>
  <genre name="Computer" copiessold="1567">
    <author name="Gambardella, Mathew" count="1">
      <book id="bk101">
        <author>Gambardella, Matthew</author>
        <title>XML Developer's Guide</title>
        <genre>Computer</genre>
        <keywords>XML software language nonfiction</keywords>
        <price>44.95</price>
        <copiessold>100</copiessold>
        <publish_date>2000-10-01</publish_date>
        <description>An in-depth look at creating applications with XML.</description>
      </book>
    </author>
    <author name="O'Brien, Tim" count="3">
      <book id="bk110">
        <author>O'Brien, Tim</author>
        ...
      </book>
      ...
    </author>
  </genre>
```

```
  <genre name="Fantasy">
    ....
  </genre>
  ...
</res:bookGenres>
```

Some points to keep in mind for nested <group> statements:

- Nested group do *not* need an `outputvariable` attribute. The results of the nested group is included in the results of its parent group.
- Each level of grouping has its *owngroup ID* to include data for the repeating items for that group. The dynamic mashup expression is in the form `{$group_id`*n*`}` for subgroups. For the second level of grouping *n* = 1, for the third level *n* = 2 and so on.

  Note: You can use `{$group_id`*n*`}`, `{$group_id`*n-1*`}` up to and including `{$group_id}` within a nested group to include repeating item data in the result.

  If the path in the `by` attribute for a nested group does not contain a nested repeating item, then the repeating item for that group is the same as the parent group. Simply put, if you are grouping on another key for the same repeating items, then `{$group_id`*n*`}` outputs the exact same items as `{$group_id`*n-1*`}` or `{$group_id}` if the parent is the top level group.

  If the path in the `by` attribute for a nested group *does* have a repeating item, this becomes the repeating item for that group. In this case, `{$group_id`*n*`}` outputs the nested level of repeating items while `{$group_id`*n-1*`}` outputs the parent level of repeating items and `{$group_id}` outputs the top level repeating items.
- The Xpath expression in the `by` attribute is *relative to the repeating group items* defined in the closest group level.

  In the previous example, books were the group items for both levels. They were grouped first by genre, then author within genre.

  You can also have several levels of group items. For example:

```
<group outputvariable="$groupedResult"
    by="$inventory//items/item/sku">
<res:items>
  <res:item id="{$group_key}" quantity="{sum(quantity)}" ...>
  <group by="parts/part/partno">
    <res:parts>
      <res:part partno="{$group_key}"
          quantity="{sum(quantity)}".../>
    </res:parts>
  </group>
  </res:item>
</res:items>
</group>
```

  The first group in this example is for items and the XPath to calculate the quantity in inventory is based on item/quantity.

The second group, however, *also* has repeating nodes in the `by` attribute - item/parts/part nodes. This becomes the group items for the second group and the quantity calculation is based on part/quantity.

- The variable `$group_key` is relative to the scope of the group that it is used in. So in the first example, the first `$group_key` generates book genres and the second generates authors.
- Calculations or the results of any XPath functions are based on the group level they are used in.

Working Samples

The following sample mashups use the <group> statement:

- GroupByService (groupby.emml) for the basics
- GroupByService (employeegroups.emml) an example of nested groups including all repeating items data
- GroupStocks (groupstocks.emml) an examle of including all repeating items data

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<sort>

This statement sorts the content of a document-type variable or variable fragment based on key expressions and places the result in another variable.

For more information and examples, see "Example 50. <sort> Basics" on page 518, "Example 51. Sort Numbers or Dates" on page 519, "Example 52. Sort With Multiple Keys" on page 519 and "Example 53. Sort Directions for Multiple Keys" on page 519. See also "Working Samples" on page 520 for information on sample mashups that use this statement.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
inputvariable	yes	The name of the input variable to be sorted.
sortexpr	yes	The path to the set of repeating nodes to be sorted.

Name	Required	Description
		This must be a literal path. Variable references and dynamic mashup expressions are *not* supported.
sortkeys	yes	The node, relative to the the sort expression path, that should be used to define sorting. This must be a literal path. Variable references and dynamic mashup expressions are *not* supported.
sortdir		The direction for sorting. Variable references and dynamic mashup expressions are *not* supported. Valid values include: ■ *ascending* = Sort is ascending. This is the default. ■ *descending* = Sort is descending.
outputvariable	yes	The required variable to accept the output of this statement.

<sort> Examples

<sort> Basics

Sorting variables is somewhat like filtering variables with a little more information:

- The input variable that should be sorted
- The specific nodes to be sorted are identified by a relative XPath expression.
- The nodes that are the key for sorting are also identified by a relative XPath expression. By default, sorting treats key values as strings. You can also "Example 51. Sort Numbers or Dates" on page 519 and "Example 52. Sort With Multiple Keys" on page 519.
- The direction of sorting (ascending is the default). You can also control the direction individually when you use multiple keys. See "Example 53. Sort Directions for Multiple Keys" on page 519 for information.
- The output variable to hold the sorted results. This can be another variable, or just update the input variable.

For example:

```
<variables>
  <variable name="queryResult" type="document"/>
  <variable name="westCoastOnly" type="document"/>
</variables>
...
<sort inputvariable="$westCoastOnly" sortdir="descending"
  sortexpr="/customers" sortkeys="salesperson"
```

```
  outputvariable="$westCoastOnly"/>
...
```

Note: The `sortexpr` is relative to the input variable and `sortkeys` is relative to `sortexpr`.

Sort Numbers or Dates

To sort numbers or dates you must explicitly cast the datatype of the sort keys to a numeric or date datatype using an XPath function. Typically, you use the constructor functions in XPath that correspond to specific datatypes in XML Schema, such as `xs:decimal` or `xs:date`.

Important: You must use `xs` as the namespace for XPath datatype constructor functions. You do not have to declare this namespace because it is built-in.

For example:

```
...
<sort inputvariable="$salesResults"
  sortexpr="region/categories/category"
  sortkeys="xs:decimal(periodSales)"
  outputvariable="$highPeriodSales"/>
...
<sort inputvariable="$troubleTickets"
  sortexpr="ticket"
  sortkeys="xs:date(created)"
  outputvariable="$troubleTickets"/>
```

Note: The constructor functions work only if the content of the sort key nodes is a valid lexical representation of that datatype. Dates, for example, must be expressed in a specific syntax. You can use other XPath functions to manipulate sort key content to obtain the correct lexical representation.

See XPath 2.0 Data Model Types for a complete list of constructor functions.

Sort With Multiple Keys

To sort with multiple keys, you add XPath expressions to the `sortkeys` attribute, separated by commas. The sort keys are used in the order that you specify them. This example sorts category nodes by category name and then by sales for each period within category:

```
...
<sort inputvariable="$salesResults"
  sortexpr="region/categories/category"
  sortkeys="@name, xs:decimal(periodSales)"
  outputvariable="$categoryPeriodSales"/>
...
```

Sort Directions for Multiple Keys

You can control the sort direction for individual sort keys when you use multiple keys by specifying the sort direction after each key. This overrides the value of the `sortdir` attribute.

For this example, tickets are sorted by creation date in descending order (most recent first) and then by customer name in ascending order:

```
...
<sort inputvariable="$troubleTickets"
  sortexpr="ticket"
  sortkeys="xs:date(created) descending, customer ascending"
  outputvariable="$troubleTickets"/>
...
```

Working Samples

The following sample mashups use the <sort> statement: rsssort, sort, sortDate

- SortSample (sort.emml) for the basics
- SortDates (sortdate.emml)
- YahooJobsSortAndFilterSample (rsssort.emml) for a filter and sort example

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<annotate>

This statement adds attributes or children nodes to node(s) of a document-type variable. Each added node is defined in text within the body of <annotate> in the form:

`[element | attribute] name value`

Both the name and value of attributes or children can be a literals or dynamic expressions. Separate multiple annotation definitions with commas.

For examples, see "Example 54. <annotate> Basics" on page 521, "Example 55. Define a New Child" on page 521, "Example 56. Define a New Attribute" on page 521, "Example 57. Set Names or Values Dynamically" on page 522, "Example 58. Define Children with Namespaces" on page 522 and "Example 59. Make Multiple Annotations to the Selected Variable and Nodes" on page 522. See also "Working Samples" on page 522 for information on mashup samples that use this statement.

Can Contain	The specific syntax for annotations.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
variable	yes	The required input variable for this statement.

Name	Required	Description
expr	yes	The relative path within the input variable to the node to be annotated.

<annotate> Examples

<annotate> Basics

You must identify the variable you want to work with and use an XPath expression to identify the specific node(s) within that variable to annotate.

```
<annotate variable="$distributionLists" expr="/lists/list" >
</annotate>
```

Then define the annotations within the body of <annotate>.

Define a New Child

Use the `element` keyword followed by the name for the new element and a dynamic mashup expression for the new element's value. New children are always appended after any existing children.

This example defines a new element with a literal name and value:

```
<annotate variable="$vendors" expr="/vendor/site" >
  element category { "metals" }
</annotate>
```

The result would look something like this:

```
<category>metals</category>
```

Note: If the parent element you are annotating has a namespace, the new element does *not* inherit this namespace unless you specify the namespace in the <annotate> statement. If you do not specify the namespace, the new child is added in the default namespace and has `xmlns=""` added as an attribute.

You can also set element names or values dynamically and set the namespace for new children. See "Example 57. Set Names or Values Dynamically" on page 522 or "Example 58. Define Children with Namespaces" on page 522 for other examples.

Define a New Attribute

Use the `attribute` keyword followed by the name for the new attribute and a dynamic mashup expression to set the attribute's value. This example adds an attribute with a literal name and value:

```
<annotate variable="$distributionLists" expr="/lists/list" >
  attribute corporate { "yes" }
</annotate>
```

Set Names or Values Dynamically

Both the name and the value for new attributes or elements can be set dynamically in Dynamic Mashup Expressions. The result of the dynamic expression must evaluate to a simple value. For example:

```
<annotate variable="$vendors" expr="/vendor/site" >
  element geo:lat { $georesult//y:Latitude/string() },
  element geo:long { $georesult//y:Longitude/string() }
</annotate>
<annotate variable="$distributionLists" expr="/lists/list" >
  attribute server { $emailServers/servers/server/name }
</annotate>
<annotate variable="$notifications" expr="/notification" >
  attribute { $property } { $colorScheme }
</annotate>
```

Define Children with Namespaces

You can also specify a namespace for new elements or attributes using the `QName(`*`namespace`*`, `*`node-name`*`)` function in the syntax for dynamic names.

This example adds an element with a literal namespace and element name:

```
<annotate variable="$payload" expr="/*:books/*:book" >
  element {QName("http://www.books.com/xmlns", "review")}
    { "Five stars!" }
</annotate>
```

The *namespace* parameter for `QName()` should be a URL or URN. If this namespace is not the same namespace as the node being annotated, the child node that is added also has an `xmlns` attribute added with this namespace.

If the parent node that you are annotating, or any ancestor, has one or more existing namespaces, you can use the `QName` function combined with the XPath `namespace-uri(`*`node`*`)` function to have the new element or attribute inherit these namespaces. This example adds a new element, with a dynamic node name and value that inherits all namespaces within the existing `result` variable:

```
<annotate variable="$result" expr="/*:records/*:record" >
  element {QName(namespace-uri($result/*), $nodeName)}
    { $newValue }
</annotate>
```

Make Multiple Annotations to the Selected Variable and Nodes

You can apply several annotations to the selected variable and targe node(s). Separate multiple annotations with commas. For example:

```
<annotate variable="$vendors" expr="/vendor/site" >
  element geo:lat { $georesult//y:Latitude/string() },
  element geo:long { $georesult//y:Longitude/string() }
</annotate>
```

Working Samples

The following sample mashups use the <annotate) statement:

- Annotation (annotate.emml) for the basics
- Annotation (annotate2.emml) for an example with a namespace
- UserDefinedFunctionSample (soundex.emml) for an example with an XPath expression

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<script>

This statement includes scripting code to execute as the mashup script is processed. Script code can be included as the content of <script> or <script> can import code from an external file deployed in the local server.

> Groovy and JavaScript are supported as scripting languages. JRuby is *deprecated* as a scripting language.

For more information and examples, see "Adding User-Defined Scripting Code to Mashups" on page 524.

Can Contain	The Groovy or JavaScript code to execute. If no inline code is provided, you must specify a source file for the code to execute.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
type	yes	The MIME type of the scripting language used, such as 'text/javascript'.
src		The local file with the scripting code to include and execute here. This is required for code that is not defined within the body of <script>.
application		An optional attribute identifying a folder containing JAR files with Java classes that are used within this script code. The MashZone NextGen Server loads classes from this application folder first. Applications allow POJO services to use third-party libraries that may conflict with MashZone

Name	Required	Description
		NextGen your Java application's libraries. Application folders must be deployed in the Application Server hosting the MashZone NextGen Server.
inputvariables		A list of mashup variables separated by commas that the script should have acccess to.
outputvariable		The mashup variable to contain the result of the script.

Adding User-Defined Scripting Code to Mashups

You can also add your own scripting code to a mashup script using the <script> statement. Your user-defined code is executed within mashup processing wherever <script> appears.

Note: You can disable the <script> command to prevent anyone from using scripts in a mashup.

Specific "Scripting Languages" on page 524 are supported in mashups. You can "Import Script Libraries" on page 525 or "Include Scripts Directly" on page 525. Scripts also have defined "Access to Mashup Variables" on page 526 and "Access to Java Classes" on page 526. See also "Adding User-Defined or Third-Party Java Classes for JavaScript Access" on page 527.

See "Working Samples" on page 527 for information on mashup samples that use scripting.

Scripting Languages

You identify the scripting language in the `type` attribute using its MIME type. EMML supports JavaScript and Groovy 1.5 as scripting languages within mashup scripts. JRuby 1.0 is supported but *deprecated*.

The setup requirements to use these languages in EMML are:

- JavaScript support is included in EMML using Rhino 1.7R4. For more information on JavaScript support, see http://www.mozilla.org/rhino/.
- Groovy support is also included in EMML. For more information on Groovy, see http://groovy.codehaus.org/.
- For JRuby support in existing mashups, you must install and configure JRuby in your development and production environments. See "Installing and Configuring JRuby for Mashups" on page 528.

  JRuby is deprecated for new development.

Import Script Libraries

You can import scripting code from JavaScript or Groovy files that are accessible from the local server using the `src` attribute in the <script> statement. To use external code, you must deploy these scripts in the MashZone NextGen Server. See "Deploying Groovy or JavaScript Scripts in MashZone NextGen" on page 528 for more information on using external scripts.

For example:

```
<script type="text/javascript" src="myFunctions.js"
  inputvariables="$order, $customer"
  outputvariable="$scriptResult"/>
...
<script type="ruby" src="myRubyScript.rb"
  inputvariables="$order, $customer"
  outputvariable="$scriptResult"/>
```

The script has access to any variables that you pass in `inputvariables`. Order or input parameters is not important. The result of the script, if any, is placed in `outputvariable`.

Include Scripts Directly

You can also simply write Groovy or JavaScript code directly in the body of <script>.

You *must* enclose your Groovy or JavaScript code in a CDATA section. This ensures that any characters in the Groovy or JavaScript code that are delimiters in XML, such as <, are not misinterpreted. For example:

```
<script type="text/javascript" inputvariables="$reviewers"
    outputvariable="$newPayload">
  <![CDATA[
    var newPayload = TopReviewers;
    if ( reviewers.reviewer.rating > 3 )  {
         newPayload.TopReviewers +=  <name>{i.name}</name>;
       }
  ]]>
</script>
...
<script type="groovy" inputvariables="$input" outputvariable="$output">
  <![CDATA[
    import groovy.xml.dom.DOMUtil
    import groovy.xml.dom.DOMCategory
    import groovy.xml.DOMBuilder
    def reader = new StringReader(input)
    def doc = DOMBuilder.parse(reader)
    def root = doc.documentElement
    use(DOMCategory) {
      def groceries = root.category.findAll{it.'@type' == 'groceries'}[0].item
      groceries.each { g ->
          g.value = 'Luxury ' + g.text()
      }
...
    }
    output = DOMUtil.serialize(root)
  ]]>
</script>
```

Access to Mashup Variables

Scripting code can *only* add to or update the mashup variable returned as the output variable of <script>.

Groovy scripts can *only* access those mashup variables that you pass to <script> using the `inputvariables` attribute. JavaScript code can access any mashup variable. However, this can impact performance. It is a best practice to pass mashup variables to JavaScript code using the `inputvariables` attribute in <script>.

You can pass any number of variables in `inputvariables`, separated by commas. For example:

```
<script type="text/javascript" inputvariables="$Orders">
<![CDATA[
  var myName = Orders.order.customer.firstname;
  var orderID = Orders.order.@id;
  var items = Orders..item \\contains all items in Orders
  for each (i in items) {
    totalprice += i.price * i.quantity;
  }
  Orders.order.item +=  <item><description>Catapult</description><price>139.95</price></item>;
]]>
</script>
...
<script type="ruby" src="myRubyScript.rb"
  inputvariables="$feedData,$myDoc" outputvariable="$rubyResult"/>
```

Access to Java Classes

For JRuby scripts in existing mashups, access to Java classes is implicit in the language. JRuby is deprecated for development in new mashups.

For JavaScript scripts, the following classes are accessible in a <script> command

- Java language classes.
- Classes in any JAR files deployed in the path identified by the `application` attribute. See "Adding User-Defined or Third-Party Java Classes for JavaScript Access" on page 527 for more information.
- Any class in the classpath for the MashZone NextGen Server.

Use fully-qualified class names, such as in this example:

```
<script type="text/javascript">
<![CDATA[
  var p = java.util.regex.Pattern.compile("a*b");
  var m = p.matcher("aaaaab");
  var b = m.matches();
  print(b);
]]>
</script>
```

Important: For Java language classes, access is dependent on the version of the JDK used by the application server that hosts the MashZone NextGen Server.

Working Samples

The following sample mashups illustrate scripting techniques:

- JavascriptSample (javascript.emml) and GoogleBlogService (datetime.emml) for JavaScript examples
- GroovySample (groovy.emml) and DynamicDataFiltering (dynamicfilter2.emml) for Groovy examples

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Adding User-Defined or Third-Party Java Classes for JavaScript Access

You can add user-defined Java classes or third party Java libraries directly to the classpath for the MashZone NextGen Server. Simply copy them to one of these folders:

- The external configuration folder, if any, for the MashZone NextGen Server.

  Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

- *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
- *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

Then, restart the MashZone NextGen Server.

Directly adding classes is simple, but can make upgrades more complicated and does not allow you to use different versions of third party libraries than the MashZone NextGen Server. To simplify migration or use specific versions of third party libraries, you add your user-define or third party classes and JARs to an *application-scope* folder.

To deploy classes in an application-scope

1. If needed, define a root folder for application scopes for the MashZone NextGen Server.

   The root folder for application scoping must reside in the same host as the MashZone NextGen Server. The default root folder for application scoping is *web-apps-home* /apps but this can be any folder. To change this default:

   a. Open the *web-apps-home* /presto/WEB-INF/classes/presto.config file for the MashZone NextGen Server in any text editor.

   b. Add the property `jems.ext.applications.dir=`*`folder-path`* with the absolute path to the folder you want to use as the root folder for all application scopes.

   c. Save your changes.

2. Create a subfolder for your application directly under the application-scope root folder (see previous steps for more information).
3. Add your classes or JARs in this subfolder and restart the MashZone NextGen Server.
4. Add `application=your-application-subfolder` to the <script> statement.

Installing and Configuring JRuby for Mashups

Note: JRuby is deprecated as a scripting language for development of new mashups. It is supported for existing mashups.

You can use JRuby as a scripting language in existing mashup scripts through the <script> statement. To use JRuby, you must install it in your environment and add configuration to MashZone NextGen.

To install and configure JRUBY

1. Download JRuby 1.0 and install it in an appropriate folder in your environment. Follow installation instructions with the JRuby libraries.
2. Add `JRUBY_HOME` as an environmental variable to the MashZone NextGen environment script:
   a. In any text editor, edit the *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat script for Windows environments or the *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh script for Mac OS X or Linux environments.
   b. Add this option to the `JAVA_OPTS` line:

      `-DJRUBY_HOME=`*path-to-jruby-installation*
3. If you have deployed the MashZone NextGen Server in WebLogic 8.1 SP6, add JRuby to the classpath:
   a. Edit the startup script for WebLogic. This is the startWebLogic.cmd or startWebLogic.sh file in the *weblogic-domain-base* folder.
   b. Add these entries to the CLASSPATH environmental variable:
      - *JRuby-home* `/lib/jruby.jar`
      - *JRuby-home* `/lib/ruby/1.8`
4. Restart the MashZone NextGen Server.

Deploying Groovy or JavaScript Scripts in MashZone NextGen

Note: JRuby is deprecated as a scripting language for mashups. You can, however, deploy JRuby scripts for mashups from previous versions.

To use Groovy or JavaScript scripts that are defined in external files with the <script> statement in mashups, you must add them to a folder that is accessible to the MashZone NextGen Server. Add the scripts to one of these locations:

- The external configuration folder, if any, for the MashZone NextGen Server.

  Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

- *web-apps-home* /presto/WEB-INF/classes for uncompressed scripts. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments. for uncompressed scripts.
- If you add scripts to a JAR, add the JAR file to *web-apps-home* /presto/WEB-INF/lib . This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

Then, restart the MashZone NextGen Server.

<xslt>

This statement processes an XSLT 2.0 stylesheet that has been deployed in the MashZone NextGen Server. You can also use XSLT 1.0 stylesheets with this statement.

There are a few Known Limitations to stylesheets using the <xslt> statement. See also "Custom XPath Functions and other XSLT Extensions" on page 531 for information on extensibility.

Before you use a mashup with <xslt>, you must Deploy the Stylesheet and Related Resources in the MashZone NextGen Server as well as any component stylesheets or additional input documents.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
script	yes	The file name of the deployed XSLT stylesheet to execute. This must be just the file name, no relative paths, folders or URLs. This can be supplied dynamically

Name	Required	Description
		with "Dynamic Mashup Expressions" on page 604.
inputvariable	yes	The document-type variable to use as the input document to be transformed.
outputvariable	yes	The required variable to accept the output of this statement. This must be a document-type variable, as only well-formed XML is supported as the output.

Known Limitations

The following XSLT features are *not* currently supported in stylesheets for the <xslt> statement:

- **Input parameters**: you cannot pass input parameters to the XSLT stylesheet with the <xslt> statement.

  A work-around to this is to add input parameters as nodes in the input document before-hand and access parameters using XPath.

- **Output methods**: only the XML method is supported. HTML, XHTML, text or custom output formats are not supported.

- **Templates matching the root node**: templates that match the root node only (<xsl:template match="/">) cause an error and stop further processing.

  The work-around to this limitation is to add the name of the document root node to the match pattern or use a wildcard such as:

  ```
  <xsl:template match="/document">
  ```

  ```
  <xsl:template match="/*[1]">
  ```

- *File Access With the doc() Function*: cannot access files through the file system with arbitrary paths or via URIs with the `file:///` protocol. XSLT styleshseets can access files:

  - With URIs using the `http://` protocol.
  - With file system paths if the file is located in the classpath.

Deploy the Stylesheet and Related Resources

Before you test or use a mashup with the <xslt> statement, you must deploy the stylesheet in the MashZone NextGen Server. You must also deploy any of the following related resources:

- Other stylesheets that are included or imported using relative paths.

- Additional input documents opened with the document() function using relative paths.

Add the stylesheet to:

- The external configuration folder, if any, for the MashZone NextGen Server.

  Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

- The *web-apps-home* /presto/WEB-INF/classesfolder. This is the default location, but is not recommended as it complicates MashZone NextGen Serverdeployments.

Add related resources in appropriate folders relative to the stylesheet, if needed.

Then, restart the MashZone NextGen Server.

Custom XPath Functions and other XSLT Extensions

You may use any custom XPath functions that you have added to the MashZone NextGen Server using the standard namespace/prefix mechanism for XSLT.

EMML uses the Saxonica 9 XSLT Engine to process XSLT stylesheets. For more information on support for XSLT extensions, please see Saxonica documentation.

Combining Component Mashable or Mashup Results

There are two ways to combine the results of two or more component mashable information sources or mashups. Joins work much like database joins. The results of the services may be very different but they have repetitive 'items' and key nodes within those items define how the items are related. See "<join>" on page 531 for instructions.

Merges, on the other hand, merge mashable or mashup results that have homogenous, document-type structures just as a database union operation does. See "<merge>" on page 535 for instructions.

<join>

This statement defines how the data from disparate variables should be joined. Variable data should have repetitive structures that are related based on some criteria in those structures - the foreign keys that define the relation. Optionally, you can also select specific nodes from the result and modify them to build the resulting document structure.

The <join> statement is comparable to inner joins for databases. Only nodes that have values that match the criteria for all variables are included.

You can obtain outer joins in mashups using XQuery. See "Using XQuery for Outer Joins" on page 634 for an example. You can also do self-joins in mashups. See "Self-Joins with a Single Dataset" on page 633 for an example.

Note: The full <join> syntax from previous releases is *no longer* supported. This syntax used <joincond> children to define join conditions.

You must define at least one join condition, as shown in Example 60. Defining Join Conditions. You can also, optionally, select which nodes to include in the joined items, as shown in Example 63. Selecting Nodes for Joined Items with <select>. See also "Example 61. Valid Join Operators" on page 534 and "Example 62. Defining Multiple Conditions" on page 534. See also "Working Samples" on page 535 for information on sample mashups that use this statement.

Can Contain	(select?)
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
outputvariable	yes	The required variable to accept the output of this statement.
joincondition	yes	An XPath 2.0 expression defining the variables to join and one or more conditions to apply to determine how variables are joined.

<select>

Optionally, defines the literal XML structure and specific nodes or computations to include in the result of the join. If this is omitted, all nodes matching the join condition are included in the result.

The structure inside <select> must be well-formed, starting with a root node, that is repeated for each included node. Elements in the structure should use a namespace, so that they are clearly separated from EMML markup. CDATA sections are allowed to contain the structure but are deprecated.

Can Contain	Can contain text and any well-formed literal XML. Data can be literal text or assigned with dynamic mashup expressions.

Attributes

Name	Required	Description
name	yes	The name to assign to the root node enclosing all selected results for the join.

<join> Examples

Defining Join Conditions

You use the `joincondition` attribute and `outputvariable` to define a basic <join> statement.

The following example is a simple join of two variables based on a single condition. For clarity, the data for the variables is shown directly. The `joincondition` attribute contains an XPath expression that defines the single condition:

```
<variable name="movies" type="document">
  <movies>
    <movie id="Stargate" ... />
    <movie id="Lassie" ... />
    ...
  </movies>
</variable>
<variable name="reviews" type="document">
  <reviews>
    <review>
     <title>Encounters of the Third Kind</title>
     <rating>4</rating>
     ...
    </review>
    <review>
     <title>Stargate</title>
     <rating>4.5</rating>
     ...
    </review>
     ...
  </reviews>
</variable>
<join outputvariable="$joinResult"
    joincondition="$movies/movies/movie/@id =
    $reviews/reviews/review/movie/title"/>
```

Each condition consists of an XPath expression, identifying a node for one item, a comparison operator and another XPath expression, identify the node for a related item to be joined. You can also use XPath functions in these XPath expressions.

Note: In earlier versions, XPath expression within the join condition did not have to contain the root node, such as `$movies/movie/@id` instead of `$movies/movies/movie/@id`. The previous syntax is *no longer* supported and will fail.

Valid Join Operators

You can use any of the basic math comparison operators (= , !=, <, <=, >, and >=).
For string comparisons, you can also use ~ to express a partial match relationship. Comparisons with ~ are not case sensitive.

You must use XML escaping with operators that include < or > characters. See "XML Escaping in URLs and Expressions" on page 600 for more information.

The following example would join an item from the `location` variable to items in the `companyProfiles` variable as long as the location city name is contained within the company profile address:

```
<join outputvariable="$joinResult"
    joincondition="$location/county/city/name ~ $companyProfiles/profiles/profile/address"/>
```

Defining Multiple Conditions

You can combine two or more conditions in the `joincondition` attribute using the logical XPath keywords `and` or `or`. You *cannot* use parentheses or other grouping symbols to define precedence, however. The `joincondition` attribute only supports very straightforward joins.

Selecting Nodes for Joined Items with <select>

The <select> element is an optional child for <join>. It allows you to define the structure of the joined items to return and selectively choose which nodes to include. If you omit <select> all the nodes from the join data are included in the output.

To define the resulting structure for the output variable:

- Define the name of the root node that encloses all results in the `name` attribute of <select>.
- Define the structure for each row or item in the result as literal XML in the body of <select>. This structure must contain:
  - The root node to wrap each repeating item from the <join> operation.
  - The descendant nodes to include within each item.

  This result structure also contains mashup expressions that determine what joined data will be selected to fill the result. See "Dynamic Mashup Expressions" on page 604 for more information.

Note: Previous versions of MashZone NextGen required you to wrap CDATA sections around the contents of <select>. CDATA is a deprecated technique. Instead, declare a result namespace on <mashup> and use this namespace to identify the output.

In the next example, the output variable will contain a <res:recommendations> element with a repeating set of <res:movie> children - the repeating items. Each <res:movie> contains a <res:movietitle> child with data from the variable named `movies` and <res:rating> and <res:comment> children with data from the variable named `reviews`.

```
<join outputvariable="$joinResult"
  joincondition="$movies/movie/@id = $reviews/review/movie/title">
  <select name="res:recommendations">
      <res:movie>
        <res:movietitle>{$movies/title}</res:movietitle>
        <res:rating>{$reviews/rating}</res:rating>
        <res:comment>{$reviews/comment}</res:comment>
      </res:movie>
  </select>
</join>
```

Working Samples

The following sample mashups use the <join> statement:

- CompanyInfoJoinSample (join.emml)
- CompanyInvokeJoinSample (joinfunctions.emml)
- JoinUsingMatches (joinmatches.emml)
- PortfolioJoine (joinselect.emml)
- ParallelSelect (multiparallel.emml)

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<merge>

Merges the results of two or more variables with a uniform structure or type. The structures of the variables must be identical, except for root node names, and they must be document type variables. Optionally, this can also select specific nodes to be included in the output.

For syndicated feeds, MashZone NextGen was configured in earlier releases to normalize feeds to a single standard format. Normalization is now turned off, by default. Instead, you can now normalize results for syndicated feeds during the merge.

For more information and examples, see "Example 64. <merge> Basics" on page 537, "Example 65. Selecting Specific Nodes from the Merged Results" on page 538 and "Example 66. Normalizing Merged Results for Syndicated Feeds" on page 538. See also "Working Samples" on page 538 for information on sample mashups that use this statement.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
inputvariables	yes	A list of comma-separated input or variable names to use as input to the operation.
outputvariable	yes	The required variable to accept the output of this statement.
select		An expression identifying the specific nodes within the merged result that should be the only nodes included in the output of this statement. This expression must define a well-formed XML structure. **Note:** This attribute is incompatible with the `feedtype` attribute.
feedtype		The standard web feed format that all merged input should be converted to. All of the merged inputs *must* be in either an RSS or Atom format for this attribute to be effective. You can 'normalize' the output to: - `rss_2.0` - `rss_1.0` - `atom_1.0` **Note:** This attribute is incompatible with the `select` attribute.
title		If the merged output is being normalized to a web feed format (the `feedtype` attribute is set), this is the title to assign to the results. **Note:** This attribute is incompatible with the `select` attribute.
description		If the merged output is being normalized to a web feed format (the `feedtype` attribute is set), this is the description to assign to the results.

Name	Required	Description
		Note: This attribute is incompatible with the `select` attribute.
author		If the merged output is being normalized to a web feed format (the `feedtype` attribute is set), this is the author to assign to the results. **Note:** This attribute is incompatible with the `select` attribute.
link		If the merged output is being normalized to a web feed format (the `feedtype` attribute is set), this is the link to assign to the results. **Note:** This attribute is incompatible with the `select` attribute.

<merge> Examples

<merge> Basics

You identify the variables to merge in `inputvariables` as a comma-separate list. You must also identify the `outputvariable` to receive the merged results.

This example shows a simple merge of results from three syndicated web feeds whose results are in that *same format*:

```
<variables>
  <variable name="YahooRSS" type="document"/>
  <variable name="FinancialNewsRSS" type="document"/>
  <variable name="ReutersRSS" type="document"/>
  <variable name="FinanceNewsUnion" type="document"/>
</variables>
...
<merge inputvariables="$YahooRSS, $FinancialNewsRss, $ReutersRSS"
  outputvariable="$FinanceNewsUnion"/>
```

When the names of root nodes are different, the MashZone NextGen Server uses the root node from the first input variable as the root for the merged output. With input variables like this:

```
<input name="data1" type="document">
  <Service1Data>
    <firstname>foo1</firstname>
    <lastname>bar1</lastname>
  </Service1Data>
</input>
<input name="data2" type="document">
  <Service2Data>
    <firstname>foo2</firstname>
    <lastname>bar2</lastname>
  </Service2Data>
```

```
</input>
<merge inputvariables="$data1, $data2" outputvariable="$result"/>
```

The result from the merge would look like this:

```
<Service1Data>
  <firstname>foo1</firstname>
  <lastname>bar1</lastname>
  <firstname>foo2</firstname>
  <lastname>bar2</lastname>
</Service1Data>
```

Selecting Specific Nodes from the Merged Results

You can also optionally set the `select` attribute with an XPath expression that identifies the specific nodes from the input variables to include in the merged results. This next example shows the use of the select attribute to include only invoice items from two database mashables:

```
<variables>
  <variable name="CorpInvoices" type="document"/>
  <variable name="DeiboldInvoices" type="document"/>
  <variable name="InvoicesUnion" type="document"/>
</variables>
...
<merge inputvariables="$CorpInvoices, $DeiboldInvoices"
  outputvariable="$InvoicesUnion"
  select="/invoices/invoice/items/item"/>
```

Normalizing Merged Results for Syndicated Feeds

When syndicated feeds have results in different standard formats, you can normalize all the inputs and then merge the results. You use the `feedtype` attribute to define the standard feed format for the results and optional `title`, `description` and `author` attributes to provide information about the result.

In this example, three web feeds in three different formats (RSS 2.0, RSS1.0 and Atom 1.0) are merged to the RSS 2.0 format:

```
<variables>
  <variable name="YahooRSS2" type="document"/>
  <variable name="FinancialNewsRSS1" type="document"/>
  <variable name="ReutersAtom" type="document"/>
  <variable name="FinanceNewsUnion" type="document"/>
</variables>
...
<merge inputvariables="$YahooRSS2, $FinancialNewsRss1,
  $ReutersAtom" feedtype="rss_2.0" title="Combined Financial News"
  description="Yahoo, Reuters and CNN financial news feeds"
  outputvariable="$FinanceNewsUnion"/>
```

Working Samples

The following sample mashups use the <merge> statement: variables

- FeedAggregationSample (merge.emml)
- UnionSample (mergenodelists.emml)
- VariableScopingRules (variables.emml)

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Constructing the Mashup Result, Inputs or Intermediate Variables

You define the structure of the result that is returned from a mashup operation in these statements:

- <constructor> to construct a well-formed document.
- <appendresult> to add one or more well-formed items to a variable.
- <select> (within <mashup>, <operation> or <macro>)
- <select> within a <join> command to select specific items within a set of repeating items.
- <group> constructs repeating structures from sorted and optionally filtered node sets.

You can also use these statements to construct complex input parameters or any complex intermediate variable. You can construct the contents of variables (in <variable>) or input parameters (in <input>) using literal XML.

Literal XML Structures with Literal or Dynamic Data

You define the literal XML structure of the result or variables. This must be well-formed XML, completely enclosing the structure in a root node.

Important: Previous versions of MashZone NextGen required you to wrap CDATA sections around literal XML in a mashup script. CDATA is a *deprecated* technique.

It is a good practice to declare a namespace on the <mashup> element for your literal XML. Then use this namespace to differentiate literal XML from EMML markup.

Namespaces for literal XML are optional. If you omit a namespace for the literal XML in your mashup script, this also limits the validation that any XML editor provides within the literal XML. It has no effect on mashup processing.

You use literal values or dynamic mashup expressions to define the data that will fill this structure. See "Dynamic Mashup Expressions" on page 604 for information on the syntax of these expressions.

You can also use XPath functions in these expressions to perform calculations or to cast data to another datatype.

<constructor>

This statement constructs a well-formed document, wrapped in a root node, and assigns it to an input or output parameter or to a variable. You define the structure as literal XML and assign static data or use dynamic mashup expressions to assign data dynamically.

> It is a good practice to use a separate namespace for the elements in the structure being constructed, so that they are clearly separated from EMML markup.

See "Example 67. Construct the Complete Document" on page 540 and "Example 68. Construct the Root and Append Repeated Content" on page 541 for examples of the two common patterns used with <constructor>. See also "Working Samples" on page 541 for information on sample mashups that use this statement.

Can Contain	Can contain text and any well-formed literal XML.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
outputvariable	yes	The required variable to accept the output of this statement.

<constructor> Examples

Construct the Complete Document

This is the basic use of <constructor> to define the entire contents of a variable. For example:

```
<mashup ...
  xmlns:svc="http://www.someservices.com/services/catalogService"
  xmlns:res="http://www.myCompany.com/myresults">
...
<constructor outputvariable="result">
    <res:mailinglist>
      <res:displayName>{$filterResult/list/name}</res:displayName>
      <res:email>{$filterResult/list/email}</res:email>
    </res:mailinglist>
</constructor>
...
<constructor outputvariable="$lookupItem">
  <svc:query>
    <svc:account>{$acctId}</svc:account>
    <svc:item>
```

```
      <svc:itemnumber>{$itemId}</svc:itemnumber>
      <svc:type>UPC</svc:type>
      <svc:scope>global</svc:scope>
    </svc:item>
  </svc:query>
</constructor>
```

The data to add can be literal text or "Dynamic Mashup Expressions" on page 604 to add dynamic data.

Construct the Root and Append Repeated Content

If you need to fill a variable with a set of repeating data, you must still wrap this in a root node. To handle this scenario, you can use <constructor> to add the root node and <appendresult> to fill in content.

For example:

```
<mashup ...
  xmlns:res="http://www.myCompany.com/myresults">
...
<constructor outputvariable="$result">
  <res:comments/>
</constructor>
...
<foreach variable="comment"
   items="$calls//customerCall/comment">
  <appendresult outputvariable="result">
  <res:comment>
     <res:customer>{$comment//custom/name/string()}</res:customer>
    <res:report>{$comment//description}</res:report>
  </res:comment>
  </appendresult>
</foreach>
```

Working Samples

The <constructor> statement is used in many of the sample mashups for MashZone NextGen. Some of particular interest include:

- ErrorHandlingSample (onerror.emml) for a basic example
- MacrosSample (macros.emml) and SOAPService (soapservice.emml) for examples of contructing requests
- LeftOuterJoinSample (outerjoin.emml) for an example of constructing and executing an XQuery query
- ParallelService (parallel.emml) for an example of merging multiple results

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<appendresult>

This statement appends the well-formed document fragment defined in its content to the specified variable. This is commonly used with repeating statements, such as <for> or <foreach>.

The document fragment contains literal XML with static values or dynamic mashup expressions. Elements in the structure should use a namespace, so that they are clearly separated from EMML markup.

For examples, see "Example 69. <appendresult> Basics" on page 542 and "Example 70. Root Nodes for Well-formed Documents from <appendresult>" on page 543. See also "Working Samples" on page 543 for information on sample mashups that use this statement.

Can Contain	Can contain text and any well-formed literal XML.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
outputvariable	yes	The required variable to accept the output of this statement.

<appendresult> Examples

<appendresult> Basics

You set the variable to be updated in `outputvariable`. With <appendresult>, this variable *must* already exist. It can be empty or already have a root node to append into. Then define the well-formed content to be added. For example:

```
<mashup ...
  xmlns:res="http://www.myCompany.com/myresults">
...
<variables>
  <variable name="result" type="document"/>
</variables>
...
<foreach variable="review" items="$reviews//amz:CustomerReviews/amz:Review">
  <appendresult outputvariable="$result">
      <res:reviewer>
         <res:name>{string($review//amz:Reviewer/amz:Name)}</res:name>
         <res:rating>{$review//amz:Rating}</res:rating>
      </res:reviewer>
  </appendresult>
</foreach>
```

The literal XML content for <appendresult> must be well formed - completely wrapped in a root node. You use literal values or "Dynamic Mashup Expressions" on page 604 to define the data that will fill this structure.

Root Nodes for Well-formed Documents from <appendresult>

If the output variable for <appendresult> has a root node already defined, <appendresult> adds repeating items as children of the root node. For example:

```
<mashup ...
  xmlns:res="http://www.myCompany.com/myresults">
...
<variables>
  <variable name="myReviews" type="document">
    <res:reviews/>
  </variable>
</variables>
<foreach variable="review" items="$reviews//amz:CustomerReviews/amz:Review">
  <appendresult outputvariable="$myReviews">
      <res:reviewer>
         <res:name>{string($review//amz:Reviewer/amz:Name)}</res:name>
         <res:rating>{$review//amz:Rating}</res:rating>
      </res:reviewer>
  </appendresult>
</foreach>
```

If the variable does not have a root node predefined, <appendresult> adds <xml> as a default root node to ensure the result is a well-formed document.

Working Samples

Several sample mashups use the <appendresult> statement. Some of particular interest include:

- DateTimeOperations (datetime.emml) for an example with a variety of XPath expressions to build the result
- GoogleFinanceNews (googlefinancenews.emml) for an example with <foreach>
- GroupByService (groupby2.emml) for an example of grouping and aggregation using <foreach> and <appendresult>
- VariableScoping (scope.emml) for an example within <parallel>

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<select>

This statement selects specific nodes within all the repeating items in an input variable and places the result in a variable. This acts as a 'column filter,' selecting specific nodes (the ’columns’) within all repeating items (the ’rows’).

For more information and an example, see "<select> Example" on page 544. See also "Working Samples" on page 545 for information on sample mashups that use this statement.

Can Contain	(columns)

Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
inputvariable	yes	The name of the input variable to select from.
selectexpr		An XPath expression that defines the set of repeating nodes in the input variable to use as the source to select from.
outputvariable	yes	The required variable to accept the output of this statement.

<columns>

The set of nodes to select from each repeating item in the results of the select expression.

Can Contain	(column+)

<column>

An XPath expression identifying one node to include in the output variable for each repeating item in the results of the select expression.

Can Contain	String

<select> Example

The variable to be filtered is identified in the `inputvariable` attribute. You identify the repeating items to filter in the `selectexpr` attribute with an XPath expression that is relative to the input variable.

The specific nodes to include in the result for this statement are defined as <column> elements inside <columns>. Each <column> element takes an XPath expression, relative to the 'items' from the select expression, to identify a node to include.

Note: Because <select> filters an existing set of nodes and does not construct a new XML document, you do *not* use dynamic mashup expressions to identify the nodes to select.

For example:

```
<select inputvariable="$feedback" outputvariable="$critical"
```

```
  selectexpr="//response[@urgency='1']">
  <columns>
    <column>respondent</column>
    <column>company</column>
    <column>description</column>
    <column>status</column>
  </columns>
</select>
```

In this example, the output of <select> is a set of <response> nodes from the $feedback variable that have an urgency value of 1. The output only includes the respondent, company, description and status nodes for each <response>.

Working Samples

The following sample mashups use the <select> statement:

- YahooHotJobSelect (select.emml)
- ParallelSelect (multiparallel.emml)

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Controlling Mashup Processing Flow

You may use several mashup statements to control the processing flow for a mashup operation. This includes:

- <if> for conditional processing.
- <for> for looping based on simple counts.
- <foreach> for looping through a set of nodes. This can be simple iterative loops or loops can be processed concurrently.
- <while> for looping as long as a condition is true.
- <break> to forcibly stop looping statements.
- <try> to catch and handle potential exceptions from statements within the <try> block.
- <throw> to forceably stop mashup processing and throw an exception as the mashup result.
- <return> to forceably stop mashup processing and return the current value of <output> .
- Concurrent Mashup Processing Flows using <parallel>.
- Database Mashable Transactions using <presto:beginTransaction>, <presto:commitTransaction> and <presto:rollbackTransaction>.
- SQL Transactions using <sqlBeginTransaction>, <sqlCommit> and <sqlRollback>.

<if>

This conditional statement follows the well-known if-elseif-else control pattern to processes specific children statements based on one or more conditions. Both <elseif> and <else> are optional.

You can use most EMML statements within <if>, <elseif> or <else>. You can also use <break> within these statements if they are descendants of a looping statement to forceably stop the ancestor loop processing.

You define the conditions that must be matched for each section of this statement to be processed as XPath 2.0 expressions. For examples, see "Example 71. <if> Basics" on page 547, "Example 72. Adding <elseif> or <else>" on page 547 or "Example 73. Breaking" on page 548. See also "Working Samples" on page 548 for information on sample mashups that use this statement.

Can Contain	(("Statements Group" on page 677 | ("Variables Group" on page 678) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (break) | (variables))+ , elseif* , else?)
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
condition	yes	An expression for the initial condition to test for the <if> statement. If this condition is matched, the statements that are direct children of <if> (not including <elseif> or <else>) are processed.

<elseif>

An optional, alternate condition with a set of statements to process if this condition is matched.

Can Contain	(Statements Group | (Variables Group) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (break) | (variables))+

Attributes

Name	Required	Description
condition	yes	An expression for the condition to test for the <elseif> statement. If this condition is matched, the statements that are direct children of <elseif> are processed and any subsequent <elseif> statements or the <else> statement is skipped.

<else>

A optional statement with a default set of children statements to process if no conditions are matched.

Can Contain	(Statements Group | (Variables Group) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (variables))+

<if> Examples

<if> Basics

At a minimum, you define <if> with a condition and add statements inside <if>, such as the following example:

```
<variables>
  <variable name="destination" type="document"/>
  <variable name="importRules" type="document"/>
  <variable name="currentCountry" type="string"/>
  <variable name="local" type="string" default="US"/>
</variables>
...
<if condition="$destination//country != $local">
  <invoke service="ImportRules" operation="getRules"
    inputvariables="currentCountry" outputvariable="$importRules"/>
</if>
```

Adding <elseif> or <else>

To include additional conditions, add <elseif> or <else> elements as children of <if>. You can have any number of <elseif> statements. For example:

```
<if condition="$sortingType/sortBy = 'date'">
  <sort intputvariable="$tasks" sortexpr="/task" sortkeys="dueDate"
    outputvariable="$tasks"/>
  <elseif condition="$sortingType/sortBy = 'name'">
    <sort intputvariable="$tasks" sortexpr="/task" sortkeys="description"
      outputvariable="$tasks"/>
  </elseif>
```

```
  <else>
    <sort intputvariable="$tasks" sortexpr="/task" sortkeys="priority"
      outputvariable="$tasks"/>
  </else>
</if>
```

Breaking

When <if> is inside a looping statement (<for>, <foreach> or <while>), you can also use <break/> inside <if> or <elseif> to stop all subsequent processing for this loop and any further looping.

For examples using <break/>, see the "<break>" on page 556 statement.

Working Samples

The following sample mashups use the <if> statement:

- FaultHandlingSample (faulthandling.emml)
- MyGooglePage (igoogle.emml)
- LevenshteinDistanceJoine (levend.emml)
- ErrorHandlingSample (onerror.emml)
- WhileSample (while.emml)

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<for>

This looping statement processes any children statements in a repeated loop for the specified count. The count may be static or may use expressions to define a dynamic limit. The increment is always one.

You can use most EMML statements within <for>. You can also use <break> within this statement to forceably stop loop processing.

Can Contain	(Statements Group | (Variables Group) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (break) | (variables))+
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
variable	yes	The name for the counter variable for this looping statement. The scope for this variable is limited to the <for> loop.
startcountervalue	yes	The number or expression that defines the number to use as the counter for the first loop. The increment is always one.
finalcountervalue	yes	The number or expression that defines the number of the last loop to execute. Expressions must evaluate to an integer value.

<for> Example

You must define the counter variable and the beginning and final values for the counter. The counter limits can be integers or XPath 2.0 expressions that evaluate to integers. The loop increment is always one.

Note: The counter variable in <for> is implicitly declared and has a local scope of the <for> command only.

As with other programming languages, you can use the counter variable name in expressions used in statements within the body of <for>.

One common use of <for> loops is to repeatedly append content to a variable, as shown in the following example:

```
<variables>
  <variable name="reviewers" type="document"/>
  <variable name="compositeResult" type="document"/>
</variables>
...
<for variable="$i" startcountervalue="1"
    finalcountervalue="count($reviewers//reviewer)">
  <display message="reviewer is " variable="$reviewers//reviewer[$i]/name"/>
  <appendresult outputvariable="$compositeResult">
     <res:ratingBy>
         <res:name>{string($reviewers//reviewer[$i]/name)}</res:name>
         <res:rating>{string($reviewers//reviewer[$i]/rating)}</res:rating>
      </res:ratingBy>
  </appendResult>
</for>
```

<foreach>

This looping statement processes any children statements in either a repeated loop or in parallel for each node in a node set. The node set is defined in an XPath expression.

You can use most EMML statements within <foreach>. You can also use <break> within this statement to forceably stop loop processing.

For examples, see "Example 74. <foreach> as Sequential Loops" on page 552, "Example 75. <foreach> as Parallel Loops for All Nodes" on page 553 or "Example 76. <foreach> as Parallel Loops for Any One Node" on page 554. See also "Working Samples" on page 554 for information on sample mashups that use this statement.

Can Contain	(Statements Group | (Variables Group) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (break) | (variables) | fuse)+
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
variable	yes	The name for the variable to hold each 'item' for this looping statement. The scope for this variable is limited to the <foreach> loop.
items	yes	An XPath 2.0 expression that defines the set of nodes (or sequence) to loop through. The length of this set defines the limit for looping and typically also defines a context for relative expressions used in statements or content within the <foreach> loop.
parallel		Determines whether each loop is processed sequentially (the default) or concurrently (`parallel="yes"`). Valid values include: ■ *yes* = Process each loop concurrently. ■ *no* = Process each loop sequentially. This is the default.
tasks		Determines how many loops are processed when loops are processed concurrently (`parallel="yes"`). This can be all loops or any one loop (the first loop to complete ends all loop processing).

Name	Required	Description
		Valid values include: ■ *InvokeAll* = Process all loops. ■ *invokeAll* = Process all loops. ■ *invokeall* = Process all loops. ■ *InvokeAny* = The completion of any one loop ends loop processing. ■ *invokeAny* = The completion of any one loop ends loop processing. ■ *invokeany* = The completion of any one loop ends loop processing.
merge		Determines how the results from multiple concurrent loops are fused into one or more variables: ■ `true` = the single output variable defined in <fuse> holds all the results of concurrent loops "by value". ■ `false` = the output variable defined in <fuse> holds a structure that references iterative variables containing the results of each concurrent loop - results "by reference". The output variable name is the base name used for iterative variables, such as $summary (the output variable), $summary1 (one loop's results), $summary2 and so on. This attribute is only meaningful when `parallel="yes"` and `tasks="invokeall"`. The order in which concurrent loops are assigned or fused in variables is not determinant.
timeout		Determines a timeout period, in seconds, for each thread when loops are processed concurrently (`parallel="yes"`).

<fuse>

Optionally, defines the literal XML structure and specific nodes or computations to include in the result when loops are processed concurrently for <foreach>. If this is

omitted, loops are processed concurrently, but the results of each loop are not bound to any variables.

How results for each loop are fused together and assigned to variable(s) depends on the value of the `merge` attribute in <foreach>. The <fuse> `outputvariable` attribute identifies the variable for the fused result, and if applicable the base name for variable names for the results of each loop.

> The structure inside <fuse> must be well-formed, enclosed in a root node. Elements in the structure should use a namespace, so that they are clearly separated from EMML markup.

Can Contain	Can contain text and any well-formed literal XML.

Attributes

Name	Required	Description
outputvariable	yes	The required variable to accept the output of this statement.

<foreach> Examples

<foreach> as Sequential Loops

For sequential loops, you must set two attributes: `items` and `variable`. You define the set of nodes to iterate through in the `items` attribute with an XPath expression. You can use predicates to control which nodes are selected, or use XPath functions as needed.

You must also identify a `variable` to hold each iteration of the set of nodes. XPath expressions for statements within <foreach> use this iteration variable to access data for the current node.

> **Note:** The iteration variable is declared implicitly and has a local scope within <foreach> only.

The following example iterates through employee nodes. Within the loop, it uses an <if> statement to test the current employee node before adding content to a variable using <appendresult>.

```
<foreach variable="staff" items="$employees_Array//employee">
  <if condition="$staff/department = 'Accounting' and
    $staff/years > 5">
    <appendresult outputvariable="$employeesResult">
       <res:vested>
           <res:name>{$staff/name)}</res:name>
           <res:vestedYears>{$staff/years - 5}</res:vestedYears>
        </res:vested>
    </if>
  </appendresult>
</foreach>
```

<foreach> as Parallel Loops for All Nodes

With concurrent loop processing, you define the iterations and variables just as you do for sequential loop processing. You set `parallel="yes"` and use the remaining attributes, plus a <fuse> child, to define how the concurrent loops are processed and how loop results are merged.

When concurrent processing is used, each node identified in the `items` attribute spawns a new concurrent task. Set `tasks="invokeall"` to ensure that each loop completes processing.

```
<foreach variable="url" items="$urls/url/string()" parallel="yes"
    tasks="invokeall" merge="true">
  <!-- invoke several web sites in parallel -->
  <directinvoke endpoint="$url" outputvariable="$tempResult"/>
  <!-- merge all results and construct final result -->
  <fuse outputvariable="$summary">
    <feedsummary title="{$tempResult/rss/channel/title}" url="{$url}">
     {$tempResult/rss/channel/item[1]/title}
    </feedsummary>
  </fuse>
</foreach>
```

With concurrent processing, you *must* construct the output of <foreach> with a <fuse> child. You define the literal XML for the result within the body of <fuse> and use static values or "Dynamic Mashup Expressions" on page 604 to define how data is assigned to the result.

The `outputvariable` for <fuse> combined with the `merge` attribute on <foreach> determines what variable(s) hold the result(s) of the concurrent loops. To aggregate all the loop results in a single variable, set `merge` to true as shown in this example.

Note: The order in which loop results are added to the combined, final result is not determinant.

The final result, assigned to the `summary` variable, for the previous concurrent <foreach> example would look something like this:

```
<taskresults>
  <taskresult>
    <feedsummary title="Yahoo! Finance"
      url="http://finance.yahoo.com/rss/headlines">
      Facebooks Boots CFO Yu, Here Comes the IPO</feedsummary>
  </taskresult>
  <taskresult>
    <feedsummary title="Business and Financial News - CNNMoney.com"
     url="http://rss.cnn.com/rss/money_topstories.rss">
     Auto bankruptcy, What it means</feedsummary>
  </taskresult>
...
</taskresults>
```

Each loop creates a <taskresult> element in which the <fuse> structure appears.

If you set the `merge` attribute to `false`, the results of each loop is assigned to its own variable. The `outputvariable` value for <fuse> is used to create individual variables as *`output-variable-name`*`1`, *`output-variable-name`*`2` and so on.

Note: The order in which loop results are assigned to individual, final result is not determinant.

The combined outputvariable contains references to each individual loop result. For the previous example, the final result assigned to the `summary` variable would look something like this:

```
<taskresults>
  <taskresult>$summary1</taskresult>
  <taskresult>$summary2</taskresult>
...
</taskresults>
```

<foreach> as Parallel Loops for Any One Node

With concurrent loops, you can also stop all further loop processing once one loop completes using `tasks` set to `invokeany`. The first loop that completes stops all other loops.

```
<foreach variable="url" items="$urls/url/string()" parallel="yes"
    tasks="invokeany" merge="true">
  <!-- invoke several web sites in parallel, stop when one returns -->
  <directinvoke endpoint="$url" outputvariable="$tempResult"/>
  <!-- construct final result -->
  <fuse outputvariable="$summary">
    <feedsummary title="{$tempResult/rss/channel/title}" url="{$url}">
     {$tempResult/rss/channel/item[1]/title}
    </feedsummary>
  </fuse>
</foreach>
```

You must still define the structure of the result using a <fuse> child and specifying the variable to hold the result. With this example, the `summary` variable might look something like this:

```
<taskresults>
  <taskresult>
    <feedsummary title="Business and Financial News - CNNMoney.com"
     url="http://rss.cnn.com/rss/money_topstories.rss">
     Auto bankruptcy, What it means</feedsummary>
  </taskresult>
</taskresults>
```

Working Samples

The following sample mashups use the <foreach> statement:

- ConcurrentInvokes (concurrent1.emml) and (concurrent2.emml)
- DynamicInvoke (dynamicinvoke.emml)
- GoogleFinanceNews (googlefinancenews.emml)
- GroupByService (groupby2.emml)
- LevenshteinDistanceJoin (levend.emml)
- GoogleWebClipping (webclipping.emml)

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<while>

This looping statement processes any children statements in a repeated loop as long as its specified condition remains true or until a <break> statement is executed. You can use most EMML statements within <while>. You can also use <break> to explictly stop further loop processing.

See also "<while> Example" on page 555 and "Working Samples" on page 556.

Can Contain	(Statements Group | (Variables Group) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (break) | (variables))+
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
condition	yes	An expression for the condition to test at the beginning or each <while> statement loop.

<while> Example

You must specify the `condition` that must be true for the <while> loop to process. Then add statements, as needed, within <while>. For example:

```
<variables>
  <variable name="orders" type="document"/>
  <variable name="subtotal" type="number"/>
  <variable name="taxes" type="document"/>
</variables>
...
<foreach variable="order" items="$orders//invoice">
  <while condition="$order/item/tax > 0">
    <appendresult outputvariable="$taxes">
       <res:order>
           <res:id>{$order/invoiceNo)}</res:id>
           <res:item>{$order/item/itemNo}</res:item>
           <res:tax>{$order/item/tax}</res:tax>
        </res:order>
    </while>
  </appendresult>
</for>
```

You can also set the condition to always be true and then explicitly break out of the loop. For an example, see the "<break>" on page 556 statement.

Working Samples

The WhileSample (while.emml) sample mashup use the <while> statement. See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<break>

This statement forceably stops loop processing for <for>, <foreach> or <while> statements. You can also use this in <if>, <elseif> or <else> statements if they are descendants of a looping statement.

You don't need to supply any additional information or content for <break/>.

Can Contain	Empty
Allowed In	elseif | for | foreach | if | while

<break> Example and Working Samples

The following example shows <break/> used inside <if> to determine when to break loop processing for a <while> statement:

```
...
<while condition="true()">
  <display message="while-loop: " expr="$cnt"/>
  <assign fromexpr="$cnt + 1" outputvariable="cnt"/>
  <if condition="$cnt > 10">
    <display message="enough already..." />
    <break/>
  </if>
</while>
```

For a working sample mashup, see the WhileSample (while.emml). See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Concurrent Mashup Processing Flows

Processing for mashup script statements is sequential by default. You can have parallel processing of mashup statements using the <parallel> statement. Then define each concurrent task in a <sequence> child within <parallel> Add variables or any mashup statement needed to complete that task inside <sequence>.

You can also have parallel processing of iterative loops using <foreach>. This is useful when the processing for each task is identical but is applied in different contexts.

<parallel>

This container statement processes one or more groups of statements in parallel. Each group of statements to be processed concurrently is defined within a <sequence> statement.

See "Concurrent Mashup Processing Flows" on page 556 for examples. See also "Working Samples" on page 557 for information on sample mashups that use this statement.

Can Contain	(sequence)+
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

<sequence>

Defines a group of statements that should be processed sequentially as one concurrent task within <parallel>.

Can Contain	(Statements Group | (Variables Group) | (presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction) | ((macro:custom-macro-name | any element in a non-EMML namespace)*) | (variables))+

Working Samples

The following sample mashups use the <parallel> statement: merge,multiparallel, parallel, scope,

- FeedAggregationSample (merge.emml)
- ParallelService (parallel.emml)
- VariableScoping (scope.emml)

See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Database Mashable Transactions

For database mashables *created in MashZone NextGen 2.7.0 or earlier*, you can add transaction handling using these MashZone NextGen extension statements:

- <presto:beginTransaction>: opens a connection to the database associated with the mashable specified in the command and sets AutoCommit to false.
- <presto:commitTransaction>: sends a commit message to the database associated with the mashable specified in the command and closes the database connection.

- <presto:rollbackTransaction>: sends a rollback message to the database associated with the mashable specified in the command and closes the database connection. The MashZone NextGen Server also automatically sends a rollback for any invocation failures for the specified Database service within the bounds of the transaction.

Important: For transaction handling for database mashables created in later versions of MashZone NextGen, see "SQL Transactions" on page 560.

The MashZone NextGen Server does not support distributed transactions. These commands only affect the invocations for a specific database mashable. All other commands are unaffected, including invocations of other services.

Transaction Example

This sample uses a database mashable with operations to insert, update or delete records in several Personnel tables.

```
...
<presto:beginTransaction service="personnel"/>
  <invoke service="personnel" operation="addEmployee"
    inputvariables="newEmployee" outputvariable="employeeResult"/>
  <invoke service="personnel" operation="addDemographics"
    inputvariables="newEmployee" outputvariable="$demoResult"/>
  <invoke service="personnel" operation="addBenefits"
    inputvariables="newEmployee" outputvariable="$benefitsResult"/>
<if condition="$employeeResult/response/response/errorcode = 300
      or $demoResult/response/response/errorcode = 300
      or $benefitsResult/response/response/errorcode = 300">
    <presto:rollbackTransaction service="personnel"/>
  <else>
    <presto:commitTransaction service="personnel"/>
  </else>
</if>
...
```

<presto:beginTransaction>

This MashZone NextGen extension statement opens a connection to a data source and marks the beginning of a database transaction within the mashup for the specified database mashable. All invocations of that service are considered part of the transaction until a rollback or commit is found.

Note: Transaction handling using this statement is only required for database services created in MashZone NextGen 2.7.0 or earlier. See "SQL Transactions" on page 560 for transaction handling information for database services created in later versions.

See "Database Mashable Transactions" on page 557 for more information and examples.

Can Contain	Empty

Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
service	yes	The name of the database mashable whose invocations should be considered part of this database transaction.

<presto:commitTransaction>

This MashZone NextGen extension statement marks the successful end and commit of a database transaction within the mashup for the specified mashable. All invocations of that service between begin and commit are considered part of the transaction. This also closes the connection to the data source.

Note: Transaction handling using this statement is only required for database services created inMashZone NextGen2.7.0 or earlier. See "SQL Transactions" on page 560 for transaction handling information for database services created in later versions.

See "Database Mashable Transactions" on page 557 for more information and examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
service	yes	The name of the database mashable whose invocations should be considered part of this database transaction.

<presto:rollbackTransaction>

ThisMashZone NextGenextension statement marks the unsuccessful end of a database transaction within the mashup for the specified mashable database. All invocations of

that service between begin and rollback are considered part of the transaction. This also closes the connection to the data source.

Note: Transaction handling using this statement is only required for database services created in MashZone NextGen 2.7.0 or earlier. See "SQL Transactions" on page 560 for transaction handling information for database services created in later versions.

See "Database Mashable Transactions" on page 557 for more information and examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
service	yes	The name of the mashable database whose invocations should be considered part of this database transaction.

SQL Transactions

You can add standard transaction handling for SQL commands invoked in a mashup script or for MashZone NextGen database mashables created in version 3.0 or later using these statements:

- <sqlBeginTransaction>: opens a connection to the default datasource or the datasource specified in the statement and sets AutoCommit to false.
- <sqlCommit>: sends a commit message to the default datasource or the datasource specified in the statement and closes the database connection.
- <sqlRollback>: sends a rollback message to the default datasource or the datasource specified in the statement and closes the database connection. The MashZone NextGen Server also automatically sends a rollback for any SQL invocation failures for the specified datasource within the bounds of the transaction.

Note: The MashZone NextGen Server does not support distributed transactions. These statement only affect the invocations for a specific datasource. All other statement are unaffected, including SQL commands to other datasources.

Transaction Example and Working Samples

```
. . .
```

```
<sqlBeginTransaction name="hr"/>
  <sqlUpdate name="hr" statement="insert into
    dept(dept_id, name, region)
    values('15', 'Analysis', 'SouthWest')"
    outputvariable="$deptInsert"/>
  <sqlUpdate name="hr" statement="insert into
    jobs(job_id, title, dept, grade)
    values('344', 'Analysis Manager', 'Analysis', 'G12')"
    outputvariable="$jobInsert"/>
  <sqlUpdate name="hr" statement="insert into
    jobs(job_id, title, dept, grade)
    values('345', 'Technical Advisor', 'Analysis', 'G15')"
    outputvariable="$jobInsert"/>
<sqlCommit name="hr"/>
...
```

For a working sample mashup, see the DatabaseSample (sql.emml). See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<sqlBeginTransaction>

This statement marks the beginning of a SQL transaction that includes all <sqlUpdate> statements until a <sqlCommit> or <sqlRollback> statement is encountered.

See also "SQL Transactions" on page 560 for more information and examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		The name of the data source for this transaction. If omitted, the transaction applies to SQL commands for the default data source.

<sqlCommit>

This statement marks the successful end of a SQL transaction that includes all <sqlUpdate> statements since the <sqlBeginTransaction> statement.

See also "SQL Transactions" on page 560 for more information and examples.

Can Contain	Empty

Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		The name of the data source for this transaction. If omitted, the transaction applies to SQL commands for the default data source.

<sqlRollback>

This statement marks the unsuccessful end of a SQL transaction that includes all <sqlUpdate> statements since the <sqlBeginTransaction> statement.

See also "SQL Transactions" on page 560 for more information and examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name		The name of the data source for this transaction. If omitted, the transaction applies to SQL commands for the default data source.

Handling or Throwing Exceptions

For mashups and macros, the default behavior when an exception is thrown from an EMML statement is to stop further processing. No result is returned.

You can use two methods to change this default behavior:

Handle Exceptions For	
Any EMML Statement	**<invoke> or <directinvoke>**
Use the <try>/<catch> statements to catch and handle exceptions thrown	With <invoke> or <directinvoke>, you can use lightweight error handling

Handle Exceptions For	
Any EMML Statement	**<invoke> or <directinvoke>**
from any EMML statement. These statements work very much like try-catch in other programming languages. See for a simple example. You define how to handle an exception with EMML statements in the <catch> block. You can use almost any EMML statement in a <catch> block, including with <throw> or with <return/>. You can use multiple <catch> blocks to catch and handle different types of exceptions based on the exception class name you specify in each <catch> block. See for more information on how exception matching is handled, examples and links to the specific exception classes that can be thrown by different EMML statements. Also see "Mashup Samples" on page 659 for working mashup samples you may try in Mashup Editor for exception handling.	outside of a <try> block by adding the `onerror` attribute. See "Lightweight Error Handling for Component Mashups and Mashables" on page 574 for more information. Or, you can add `onerror` to <invoke> or <directinvoke> inside a <try> block to override any <catch> blocks. See for more information.

You can also use both the <throw> statement and the <return> statement anywhere in a mashup or macro. Use <throw> to create and throw a custom exception for a mashup or macro. Use <return> to forcibly stop mashup or macro processing, but still return a result.

<try>

This block statement follows the well-known try-catch control pattern to process children statements and catch and handle any exceptions thrown by those statements. You must specify at least one <catch> statement.

You can use most EMML statements within <try> or <catch>. Some of the most common are statements, such as <invoke>, <directinvoke>, <sql>, <script> or <xslt> that retrieve data or use other programming languages.

You can also use <throw> within <try> or <catch> to throw exceptions, and possibly stop further mashup or macro processing. Or use <return> to forcibly stop further mashup or macro processing and return the current value of <output>.

For examples, see:

- "Example 79. A Basic <try>/<catch> Block" on page 565
- "Example 80. Throwing Custom Exceptions in <catch>" on page 566
- "Example 81. Forcibly Stopping Processing in <catch>" on page 567
- "Example 82. Exception Propagation" on page 567
- "Example 83. Exception Matching for <catch> Blocks" on page 568
- "<directinvoke> Exceptions" on page 569
- "<invoke> Exceptions" on page 570
- "<script> Exceptions" on page 571
- "Specific Java Exceptions for <sql>, <xslt>, XPath or Syntax Exceptions" on page 571
- "Default EMML Exception Class" on page 571

.

Can Contain	((Statements Group | Variables Group | Declarations Group | Macroincludes Group| presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction | macro:custom-macro-name | (any element in a non-EMML namespace))+ followed by (<catch>)+)
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

<catch>

A block to catch and handle a specific type of exception thrown by statements in the parent <try> block.

Can Contain	((Statements Group | Variables Group | Declarations Group | Macroincludes Group| presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction | macro:custom-macro-name | (any element in a non-EMML namespace))+)
Allowed In	try

Attributes

Name	Required	Description
type	yes	A string in the form: *exception-class-name* `[`*variable-name* `]` Where *exception-class-name* is required and identifies the exception class this <catch> block handles. Exception names are *not* case sensitive in this context. See "Example 83. Exception Matching for <catch> Blocks" on page 568 for more information on valid exception names in various contexts and how <catch> blocks are matched. The second component, *variable-name* , is optional and identifies the variable name you assign to the exception. If you omit *variable-name* , `$exception` is the default variable name for the exception.

Examples

A Basic <try>/<catch> Block

The following example is a minimal <try>/<catch> block that executes an invalid SQL query inside <try> and catches any exception.

```
<mashup name="SimpleTryCatch"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd" xmlns="h
  <output name="result" type="document"/>
    <try>
      <!-- invalid query -->
      <sql name="MyDataSource" query="select * from titles"
          outputvariable="result" />
      <catch type="EMMLException">
        <display message="exception thrown: " expr="$exception"/>
        <!-- do nothing else-->
      </catch>
    </try>
</mashup>
```

With this example:

- The mashup response is not assigned any data because the SQL query fails.
- The <catch> block handles any exception thrown by the <sql> statement because `EMMLException` is the "Default EMML Exception Class" on page 571 for any EMML statement. The exception is assigned to the default variable `exception`.

> **Note:** You can also provide your own variable name to use for the exception. In `<catch EMMMMLException e">` the variable for the exception will be `$e`.

- The <display> statement in <catch> uses the default variable `$exception` to include the entire exception in the message.

  Depending on the source of the exception, exceptions may be simple strings or well-formed XML documents most commonly in one of these forms:

```
<exception-class-name>
  <errorcode>nnnn</errorcode>
  <message>description of the exception</message>
</exception-class-name>
```

  or

```
<exception-class-name>
    <message>description of the exception</message>
</exception-class-name>
```

  or

```
<exception-class-name>description of the exception</exception-class-name>
```

  But other forms occur based on the context of the exception. For example, exceptions thrown by WSDL mashables follow SOAP standards.

- The mashup returns a success response of `<xml/>` because the output variable is empty.

See also "Example 80. Throwing Custom Exceptions in <catch>" on page 566, "Example 81. Forcibly Stopping Processing in <catch>" on page 567 and "Example 83. Exception Matching for <catch> Blocks" on page 568 for more examples and information.

Throwing Custom Exceptions in <catch>

In some cases, you may want to throw a different exception from a <catch> block to provide other information or a user-friendly error message. This uses the <throw> statement, literal XML and a dynamic mashup expression to represent the exception. For example:

```
<mashup name="TransposeResultsMashup" ... >
...
  <try>
    <xslt inputvariable="$myDoc" script="transpose.xsl" output="result"/>
    <catch type="javax.xml.transform.Transformer e">
      <variable name="msg" type="string"/>
      <assign fromexpr="concat("Your input document was not transposed successfully. The error wa
<throw>
<XsltTransformException context="myOrg.TransposeResultsMashup">
<message>{$msg}</message>
</XsltTransformException>
</throw>
    </catch>
  </try>
...
</mashup>
```

Unless this <try> block also contains <catch type="XsltTransformException">, this new exception will be thrown by the mashup and processing stops if the <xslt> statement fails to complete the transformation.

This exception provides both a more specific exception name and some additional context that can help track down the error, such as the organization and mashup name in the `context` attribute. As well, the message itself is more user friendly that the original Java exception.

Forcibly Stopping Processing in <catch>

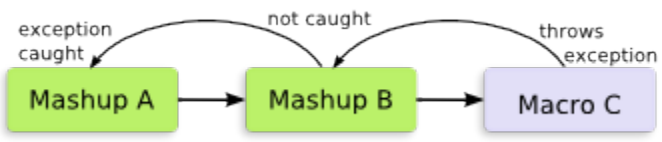
With some exceptions, you want to stop further mashup or macro processing but still return a result. You can do this with the <return> statement. For example:

```
...
  <output name="result" type="document"/>
  ...
  <foreach variable="member" items="$selectedMembers//member/name">
<try>
<variable name="newList" type="string"/>
<macro:addToFormattedList list="$partial" newItem="$member"
outputvariable="newList"/>
<catch type="NoNewListItem e">
<!-- thrown by macro when newItem is empty -->
<constructor outputvariable="$result">
<members>{$partial}</members>
</constructor>
<return/>
</catch>
</try>
  </foreach>
...
</mashup>
```

This example calls a macro in the <try> block to assemble a formatted list of member names. If the macro throws a custom exception named `NoNewListItem`, because the member name that was passed was empty, the <catch> block constructs a result for the mashup using the partial formatted list already assembled and calls <return/> to stop processing and return this partial list.

Exception Propagation

When mashups invoke other mashups or call macros, exceptions can propagate up through each level of mashup or macro until they are caught. For example:



The exception thrown in this example by Macro C propagates up the call stack (through Mashup B) to Mashup A where it is finally caught.

Exception Matching for <catch> Blocks

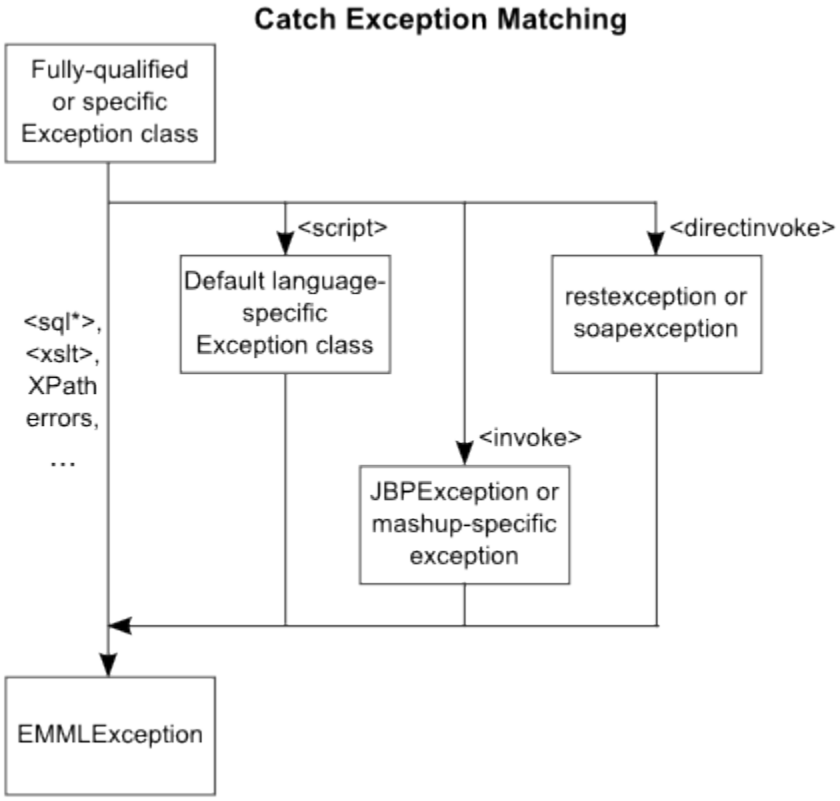
You identify what exceptions each <catch> block will handle in the `type` attribute with an exception class name. The name can be either:

- Fully-qualified class names, such as `java.sql.SQLException`
- Just the class name, such as `SQLException` or `sqlexception`

```
...
  <try>
    ...
<catch type="SQLException e">
      <!-- create error to contact DB administrator -->
    </catch>
<catch type="java.net.ConnectException e">
      <!-- create error to contactMashZone NextGenadminstrator -->
    </catch>
<catch type="emmlexception e">
      <!-- default catch, do nothing -->
    </catch>
  </try>
...
```

Matching exception names to a <catch> block is not case sensitive.

With multiple <catch> blocks, the order of <catch> blocks does not determine which block handles a given exception. Instead, exceptions are handled by the <catch> block with the exception name that is the most *specific* match.



The potential exception classes that can be thrown in a <try> block depend on the context of the exception:

- <directinvoke> Exceptions
- <invoke> Exceptions
- <script> Exceptions
- Specific Java Exceptions for <sql>, <xslt>, XPath or Syntax Exceptions
- Default EMML Exception Class

<directinvoke> Exceptions

For <directinvoke> within a <try> block, the `onerror` attribute affects how exceptions are treated. If `onerror="continue"` within <try>, all <catch> blocks are *ignored*. The exception is handled just as it is when <directinvoke> with `onerror="continue"` is outside a <try> block. See "Lightweight Error Handling for Component Mashups and Mashables" on page 574 for more information.

If `onerror` is not included or `onerror="abort"` is used, then you can specify one of the following EMML exception classes or use the Default EMML Exception Class. These EMML exception classes are used for responses from the endpoint based on the HTTP status code and the content of the response:

- *restexception* is used for responses from REST web services, syndicated web feeds or web sites (HTML) when the HTTP status code is in the 300s, 400s or 500s. For example:

```
...
  <try>
<directinvoke endpoint="http://localhost/foo302.php"
outputvariable="result" followredirects="false"
responseheader="$responseHeader" responsecode="$responseCode"/>
<catch type="restexception e">
      <if condition="$responseCode >= 500 ">
        <display message="caught restException 5xxx " />
        <elseif condition="$responseCode >= 400 ">
          <display message="caught restException 4xxx " />
        </elseif>
        <elseif condition="$responseCode >= 300 ">
          <display message="caught restException 3xxx " />
        </elseif>
      </if>
</catch>
  </try>
...
```

- *soapexception* is used for responses from WSDL web services when the response has an HTTP 500 status code and the response contains a SOAP fault for SOAP 1.1 or 1.2.

  Note: If the response contains a SOAP fault, but does not use the HTTP 500 status code, this is not considered a SOAP exception (per the Web Service standard).

  For example:

```
...
  <try>
<directinvoke endpoint="http://localhost/soapfault1.php"
outputvariable="result1" />
<catch type="soapexception se">
      <display message="caught soapException= " expr="$se//*:Fault"/>
      <display message="soapException Value = " expr="$se//*:Value/string()"/>
</catch>
  </try>
...
```

<invoke> Exceptions

For <invoke> within a <try> block, the `onerror` attribute affects how exceptions are treated. If `onerror="continue"` within <try>, all <catch> blocks are *ignored*. The exception is handled just as it is when <invoke> with `onerror="continue"` is outside a <try> block. See "Lightweight Error Handling for Component Mashups and Mashables" on page 574 for more information.

If `onerror` is not included or `onerror="abort"` is used, you can use the *JBPException* class or use the Default EMML Exception Class.

<script> Exceptions

In most cases, it is a best practice to handle exceptions from <script> within the scripting code itself. If this is not possible, the potential exception classes for <script> depend on the scripting language in use:

JavaScript	`RhinoException` or `org.mozilla.javascript.RhinoException` for scripts in JavaScript.
Groovy	`GroovyRuntimeException` or `groovy.lang.GroovyRuntimeException` or any subclass for scripts in Groovy. See http://groovy.codehaus.org/api/groovy/lang/GroovyRuntimeException.html for a complete list.

Otherwise, use the Default EMML Exception Class.

Specific Java Exceptions for <sql>, <xslt>, XPath or Syntax Exceptions

For other EMML statements, you must specify a specific Java exception or use the Default EMML Exception Class. Some of the most common exceptions include:

<sql>, <sqlUpdate> or <sqlcall>	`java.sql.* exceptions` for the <sql>, <sqlUpdate> or <sqlcall> statements.
<xslt>	`javax.xml.transform.* exceptions` or `net.sf.saxon.* exceptions` for the <xslt> statement.
XPath errors for any statement	`XPathException`, `XPathExpressionException` or `XPathFunctionException` for XPath errors from any EMML statement.
Parser errors	For syntax errors.

Default EMML Exception Class

The default exception class for all EMML statements is `EMMLException` or `org.oma.emml.me.commons.EMMLException`. It is always the least specific match to an exception. Most commonly, you use this to catch any exceptions not caught by other <catch> blocks. See "Example 79. A Basic <try>/<catch> Block" on page 565 for an example.

<throw>

This statement constructs an exception from its content and throws this exception. If the exception is not caught in a <catch> block within the mashup or macro, this forcibly stops all further mashup or macro processing.

You define the exception as well-formed, literal XML inside <throw> and assign static data or use dynamic mashup expressions to assign data dynamically. See "Creating Custom Exceptions" on page 572 for more information and best practices.

This statement is most commonly used inside a <catch> block within the <try> statement. You can, however, use it anywhere in a mashup or macro, in looping statements or in other control statements. See "Example 84. Examples" on page 573.

Can Contain	Can contain text and any well-formed literal XML. See "Creating Custom Exceptions" on page 572 and "Example 84. Examples" on page 573 for more information.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Creating Custom Exceptions

Custom exceptions are simply well-formed XML documents in this general form:

```
<exception-classname optional-context-attribute>
  <errorcode>number or code</errorcode>
  <message>description</message>
</exception-classname>
```

The exception *must* contain at least a root node. The name of this root node defines the class of the exception and must be a valid XML name.

Tip: It is a good practice when you throw a custom exception from a mashup or macro to use your own unique exception names *rather than* using existing exception class names, such as Java exceptions. Reusing exception class names can be confusing and make it more difficult to resolve problems if the exception does occur.

The exception can contain any of the following optional children:

- <errorcode> for a numeric or string code that identifies the error which has occurred
- <message> with the description of the error
- Any other child elements to contain information pertinent to the exception.
- An optional attribute of any name on the root node of the custom exception with additional context information such as your organization name, the module or

application where this occurred and the name of the mashup or macro that threw this exception.

> **Tip:** It is a best practice to add this additional context as it can help track errors down. This acts much like the fully qualified class name of a Java exception.

The examples in this topic use an attribute named `context`, but you can use any name you want.

Examples

You can throw custom exceptions to provide better information from <catch> blocks. See "Example 80. Throwing Custom Exceptions in <catch>" on page 566 for an example.

You can also simply rethrow an exception from a <catch> block:

```
<try>
  <macro:getAdjustedCost height="$this.height" width="$this.width"
      outpuvariable="$adjustedCost"/>
  <catch type="IllegalArgument">
    ...
  </catch>
<catch type="EMMLException">
<assign literal="1" outputvariable="$defaultWidth"/>
<assign literal="1" outputvariable="$defaultHeight"/>
<!-- and rethrow this exception -->
<throw/>
  </catch>
</try>
```

But you can throw exceptions anywhere within a mashup or macro, such as this example:

```
...
  <input name="maximumWidth" type="number"/>
  <input name="maximumHeight" type="number"/>
  <input name="unit" type="string"/>
<if condition="not(matches($unit,'ft') or matches($unit, 'yd'
or matches($unit 'm'))">
<throw>
<InvalidMeasurement context="myOrg.adjustEstimateMashup">
<message>The measurement unit must be feet, yards or meters.</message>
</InvalidMeasurement>
</thow>
</if>
...
```

<return>

This statement forcibly stops all further mashup or macro processing and returns the current value of <output> as the mashup or macro result.

You don't need to supply any additional information or content for <return/>.

For examples, see "Example 81. Forcibly Stopping Processing in <catch>" on page 567 and "Example 85. Forcibly Stopping Mashup or Macro Processing" on page 574 for examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Forcibly Stopping Mashup or Macro Processing

This example checks to ensure that filter results contain at least one row. If not, it constructs a result with an error message and uses <return/> to stop all further processing.

```
...
  <output name="result" type="document"/>
  <!-- invoke mashable and filter results to $rows variable-->
  <if condition="empty($rows)">
    <variable name="msg" type="string"/>
    <assign fromexpr="concat('No rows meet your criteria: ',$filterBy)"
      outputvariable="msg"/>
<constructor name="result">
<rows>
<error>{$msg}</error>
</rows>
</constructor>
<return/>
  </if>
...
```

Lightweight Error Handling for Component Mashups and Mashables

When mashups or macros invoke a mashable information source or another mashup, you can handle exceptions that may be thrown from those invocations using a <try> block, just as you can for any EMML statement.

Or instead, you can use the `onerror` attribute in <directinvoke> or <invoke> as lightweight exception handling *outside* of a <try> block. See "Controlling Component Mashup/Mashable Invocation at Runtime" on page 491 for specific information.

If you set `onerror="continue"` for exceptions from a component mashup or mashable that is not wrapped in a <try> block, the MashZone NextGen Server continues mashup or macro processing after an exception occurs and returns error information in two built-in variables:

- **faultcode** =
  - -2 for timeout errors.
  - -1 for invocation errors. In this case the error message is also returned.
  - 0 for successful invocations.
- **faultmessage** = the error message from the exception, if this was a mashup or mashable error.

Note: You may also see variables named *fault* or *faultexception*. The fault variable is reserved for future use. The faultexception variable is for internal use.

This example detects the exception with <if>, constructs the mashup result to indicate what error has occurred and then uses "<return>" on page 573 to stop further execution:

```
  <invoke service="$service" operation="getFeed"
      outputvariable="$result" onerror="continue"/>
<if condition="$faultcode = -1">
    <!-- invocation exception -->
    <constructor outputvariable="$result">
      <Error>{$faultmessage}</Error>
    </constructor>
    <return/>
<elsif condition="$faultcode = -2">
      <!-- timeout -->
      <constructor outputvariable="$result">
        <Error>Service invocation timeout</Error>
      </constructor>
      <return/>
    </elsif>
<elseif condition="$faultcode = 0">
      <!-- success, do nothing -->
      <display message="success ..." expr="$faultmessage" />
    </elseif>
  </if>
```

Supporting Debugging

You can use the following EMML statements to help debugging mashup scripts:

- <display> to send debugging messages to the console and standard output. This can include static text and values from variables.
- <assert> to include assertions tested at runtime that stop mashup processing for failures.

<display>

This statement sends a debugging message to the standard output (console and optionally logs) when the mashup script is processed. Messages can contain both static and dynamic content.

See also "<display> Examples" on page 576 and "Working Samples" on page 577.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
message	yes	The static portion of a debugging message for this mashup script.
expr		An optional XPath 2.0 expression to provide a dynamic value as part of a debuggning message for this mashup script. You cannot have both an expression and a variable in a debugging message.
variable		An optional variable to provide a dynamic value as part of a debuggning message for this mashup script. You cannot have both an expression and a variable in a debugging message.
console		An optional flag to control logging for the output for this statement. By default, <display> output is *only* logged when the logging level for the application server is set to INFO or lower. When testing from command line utilities or from tool testing views (run mode), output appears in the console *after* the mashup has run. Add this attribute with a value of `true`, to log output from <display> as it is processed when testing from the command line or in testing views (run mode) in tools.
type	no	Type attribute value must be 'stream' or 'document'. To display EMML stream variables use 'stream' type.
previewcount	yes, when type='stream'	Indicates number of records in stream variable to be displayed. It must be a positive integer count or 'ALL'.

<display> Examples

You define static text for the message in the `message` attribute. You can also optionally include a dynamic value at the end of the message with either the `expr` attribute and

an XPath 2.0 expression or with the `variable` attribute which points to a variable. For example:

```
<display message="testing debugging messages"/>
<display message="filtered result = " variable="$result"/>
<display message="sum = " expr = "$firstNum + $secondNum"/>
```

Output always appears in the console. However, logging the results of <display> is controlled by the logging level of the application server hosting the MashZone NextGen Server. Output for <display> is only logged if the level is INFO or lower.

To force logging for <display> statements, add the `console` attribute:

```
<display console="true" message="sum = " expr="$firstNum + $secondNum"/>
```

If you want to turn off debug messages for production, enclose the <display> commands in XML comments, such as:

```
<!-- <display message="debug message"/> -->
```

Or remove them from production mashup scripts.

EMML 'stream' variables can be displayed using display statement with type='stream' and previewcount='<n> or ALL' attributes.

Sample uses,

```
<display message="display 2 records in mfgStream = "
            type="stream" variable="$mfgStream" previewcount="2"/>
<display message="display 5 records in mfgStream = "
            type="stream" variable="$mfgStream" previewcount="5"/>
<display message="display all records in  mfgStream = "
            type="stream" variable="$mfgStream" previewcount="all"/>
```

Working Samples

Most of the sample mashups for MashZone NextGen use the <display> statement. See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

<assert>

Use <assert> to add error checking assertions within a mashup, operation, macro or other statement. Assertions are evaluated at runtime and throw exceptions if they evaluate to false.

Assertions may compare any two of the following:

- A variable
- An XPath expression of any kind
- A literal value
- A count of nodes identified by an XPath expression
- A count of the levels of descendants from a node or node set identified by an XPath expression

Assertion comparisons for variables or expressions that result in simple types check both the datatype and the value. For variables or expressions that are a document type (complex), the comparison checks for node-by-node equality both in node names and values. Whitespace, however, is not considered in document-type comparisons.

When assertions fail, they throw an `AssertionFailedException` with appropriate messages. For more information and examples, see "Example 86. Assertions for Simple Types" on page 581, "Example 87. Assertions for Document Types" on page 581 or "Example 88. Counts and Element Depths in Assertions" on page 581. See also "Working Samples" on page 581 for information on sample mashups that use this statement.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
actualexpr		An XPath expression identifying one or more nodes, values or any valid calculation to be used in the comparison for this assertion. Assertions must contain at least one of `actualexpr` or `actualvar`.
actualvar		The variable to be used in the comparison for this assertion. For variables that are a document type, the comparison is made for all nodes. Assertions must contain at least one of `actualexpr` or `actualvar`.
equals		Optionally determines what operator is used for the comparison of this assertion. True (the default) uses equals as the operator or false uses not equals.
expectedexpr		An XPath expression identifying one or more nodes, values or any valid calculation that the `actualexpr` or `actualvar` should be compared to. Assertions must contain at least one of `expectedexpr`, `expectedvar`, `literal`, `count` or `elementdepth`.

Name	Required	Description
expectedvar		The variable that the `actualexpr` or `actualvar` should be compared to. Assertions must contain at least one of `expectedexpr`, `expectedvar`, `literal`, `count` or `elementdepth`.
literal		The literal value that the `actualexpr` or `actualvar` should be compared to. This value is treated as a string. Assertions must contain at least one of `expectedexpr`, `expectedvar`, `literal`, `count` or `elementdepth`.
count		An XPath expression identifying a set of one or more nodes. The `actualexpr` or `actualvar` is compared against the number of nodes found in this node set. Assertions must contain at least one of `expectedexpr`, `expectedvar`, `literal`, `count` or `elementdepth`.
elementdepth		An XPath expression identifying a set of one or more nodes. The `actualexpr` or `actualvar` is compared against the number of levels of descendants *plus 1* (for the node itself) of the nodes in this node set. Assertions must contain at least one of `expectedexpr`, `expectedvar`, `literal`, `count` or `elementdepth`.

<assert> Examples

All the assertion examples in this section are based on the following variable definitions:

```
<variable name="emailA" type="string" default="nwatts@myorg.com" />
<variable name="emailB" type="string" default="nwatts@myorg.com" />
<variable name="boolA" type="boolean" default="true"/>
<variable name="numberA" type="number" default="3"/>
<variable name="doc1" type="document">
  <matches>
    <match>
      <team1>India</team1>
      <team2>Bermuda</team2>
      <venue>Trinidad</venue>
    </match>
    <match>
      <team1>India</team1>
      <team2>Sri Lanka</team2>
```

```
      <venue>Port of Spain</venue>
    </match>
    <match>
      <team1>England</team1>
      <team2>Canada</team2>
      <venue>St Lucia</venue>
    </match>
  </matches>
</variable>
<variable name="doc2" type="document">
  <bets>
    <odds>
      <for>15</for>
      <against>30</against>
    </odds>
    <odds>
      <for>25</for>
      <against>20</against>
    </odds>
    <odds>
      <for>18</for>
      <against>30</against>
    </odds>
  </bets>
</variable>
<variable name="doc3" type="document">
  <odds>
    <for>18</for>
    <against>30</against>
  </odds>
</variable>
<variable name="doc4" type="document">
  <tracking>
    <stats team="Bermuda">
      <wins>3</wins>
      <losses>5</losses>
      <past-div-winner>false</past-div-winner>
      <good-reviews>
        <newspaper>3</newspaper>
        <magazine>0</magazine>
        <online>5</online>
      </good-reviews>
    </stats>
    <stats team="India">
      <wins>6</wins>
      <losses>2</losses>
      <past-div-winner>true</past-div-winner>
      <good-reviews>
        <newspaper>4</newspaper>
        <magazine>1</magazine>
        <online>7</online>
      </good-reviews>
    </stats>
  </tracking>
</variable>
<variable name="doc5" type="document">
  <matches>
    <match>
      <team1>England</team1>
      <team2>Canada</team2>
      <venue>St Lucia</venue>
    </match>
    <match>
```

```
      <team1>India</team1>
      <team2>Sri Lanka</team2>
      <venue>Port of Spain</venue>
    </match>
  </matches>
</variable>
```

Assertions for Simple Types

The first assertion in this example compares a variable to a literal and expects the result to be false. The second assertion compares two variables that are simple string values with an expected result to true.

```
<assert actualvar="$numberA" literal="10" equals="false"/>
<assert actualvar="$emailA" expectedvar="$emailB" equals="true"/>
<assert actualexpr="$doc4/tracking/stats[@team='India']/past-div-winner" expectedvar="$boolA" equ
```

The last assertion compares an expression that resolves to one node with a simple value within a document-type variable to a variable with a simple type.

Assertions for Document Types

The first document-type assertion compares one expression to a variable. Every node in the variable must be present and equal in both name and value in the node set defined by the expression and vice versa.

```
<assert actualexpr="$doc2/bets/odds[last()]" expectedvar="doc3" equals="true"/>
<assert actualexpr="$doc1/matches/match[last()]" expectedexpr="$doc5/matches/match[position()=1]"
```

The next document-type assertion compares two expressions which both result in node sets. Again, all nodes from the first expression must be present and equal to the nodes in the second expression and vice versa.

Counts and Element Depths in Assertions

Assertions using count are fairly simple, as the first example assertion shows. The number of nodes in the variable or expression being compared are counted and compared against the value of the `count` attribute.

```
<assert actualexpr="$doc2/bets/odd" count="3" equals="true"/>
<assert actualexpr="$doc4/tracking/stats" elementdepth="3" equals="true"/>
```

With element depth, the number of levels of descendants found in the variable or expression being compared *including the root level* of the variable or expression is compared to the value of the `elementdepth` attribute.

Working Samples

The AssertSample (assert.emml) sample mashup uses the <assert> statement. See "Mashup Samples" on page 659 for a list of MashZone NextGen mashup samples and where to find them.

Adding Metadata to Mashups

Metadata declarations allow you to set processing flags or pass other information to applications that will use a mashup, use a macro or handle mashup results. Metadata is

stored with mashups in the MashZone NextGen Repository and can be queried Using the Built-in Mashup Query Web Service.

The metadata declarations that you can use include:

- <emml-meta> for metadata built into EMML. It provides processing flags or other information used by the MashZone NextGen Server.
- <presto:macro-meta> for metadata for macros that is used by MashZone NextGen. It provides processing flags used by the MashZone NextGen Server or by mashup composers such as Wires.

  Note: <presto:macro-meta> is a new metadata element that replaces variations of the <presto:presto-meta> element specific to macros.

- <presto:presto-meta> for metadata for mashups that is used by MashZone NextGen. It provides processing flags used by the MashZone NextGen Server or by mashup composers such as Wires.

  Note: Variations of <presto:presto-meta> specific to macros have been *deprecated*. Use <presto:macro-meta> instead.

- <user-meta> to define your own custom metadata. You can use this for flags or any other data you wish to associate with a mashup or a macro. You define the names and content of user metadata to fit your requirements.

Metadata Names and Values

Most metadata declarations have a `name` attribute that identifies the purpose of the metadata. Names for user-defined metadata must:

- Be unique.
- Start with an ASCII letter and contain only ASCII letters, numbers, dashes (-) or underscores (_). Names are case sensitive.

Metadata declarations can contain a simple value or a complex structure of descendant elements. Simple values for metadata are a string datatype.

For complex structures, descendants *can only* be your own custom metadata or extensions specifically allowed in <presto:presto-meta> or <presto:macro-meta>. Other EMML statements or declarations are not allowed.

For example:

```
<user-meta name="refresh-seconds">5</user-meta>
<presto:presto-meta name="created-using">Wires</presto:presto-meta>
<presto:macro-meta>
  <block usage="Wires">
    <help>Returns all open issues for the customer specified as input.</help>
    <icon usage="menu" url="http://myOrg.com/icons/myBlockSmall.png"/>
    <icon usage="canvas" url="http://myOrg.com/icons/myBlockMed.png"/>
  </block>
  <parameters>
    <parameter name="query">
      <label>Customer</label>
```

```
      <help>Enter part or all of a customer name to find open issues.</help>
      <type datatype="text"/>
    </parameter>
  </parameters>
</presto:macro-meta>
```

<emml-meta>

One EMML built-in metadata property to assign to this mashup or macro.

There is currently only one built-in metadata statement for EMML to identify the author of a mashup. For example:

```
<emml-meta name="author">Karl Nguyen</emml-meta>
```

See also "Adding Metadata to Mashups" on page 581 for more information and examples.

Can Contain	Metadata for this mashup using built-in metadata statements for EMML.
Allowed In	mashup | macro | operation

Attributes

Name	Required	Description
name	yes	`author` is the only valid value. This defines the author of this mashup or macro.

<presto:macro-meta>

This metadata element defines configuration that MashZone NextGen uses to handle macros defined in macro libraries. It determines whether the macro should be used as a custom block in Wires, and if so how properties should be defined to capture the input for this macro.

Note: This metadata element replaces the following *deprecated* uses of the <presto:presto-meta> element from previous releases:

- `<presto:presto-meta name="help">`
- `<presto:presto-meta name="macrotype">`
- `<presto:presto-meta name="type">`

If you *omit* <presto:macro-meta>, MashZone NextGen treats macros as custom blocks in Wires and uses defaults to render the block and provide a block properties form. See "Default Custom Block With No Configuration" on page 404 for more information.

Can Contain	<block>, <parameters>?
Allowed In	<macro>

<block>

This element determines whether or not a macro is available as a custom block in Wires. For macros that define custom blocks, this element contains configuration that applies to the entire block.

Can Contain	<label>?, <help>?, (<icon>, <icon>)?, <uiconfig>?
Allowed In	<presto:macro-meta>

Attributes

Name	Required	Description
usage	yes	■ `Wires` = this macro defines a custom block. ■ `System` = this macro does not define a custom block.
resourcebundle		Reserved for future use.

<label>

The label to display in Wires in the Blocks Menu, for this block, or in the Block Properties pane, for this input parameter. If omitted, Wires displays the macro name as the label for the block and input parameter names for the corresponding block properties.

Can Contain	Any text that is valid in XML.
Allowed In	<block> | <parameter>

Attributes

Name	Required	Description
key		Reserved for future use.

<help>

User assistance, in several forms, for the custom block for this macro. You can provide three types of user assistance:

- A tooltip for the block that appears in the Wires Blocks menu.
- A tooltip for an input parameter that appears as a property in the Wires Block Properties pane.
- A help topic, written as an HTML page, that opens from the **?** button in the Wires Block Properties pane.

See "Add Tooltips and Help in Custom Blocks" on page 409 for examples.

Can Contain	Any text that is valid in XML for the tooltip for this block or this parameter, depending on the context for this element.
Allowed In	<block> | <parameter>

Attributes

Name	Required	Description

url		An absolute or relative URL to the HTML page that is the help topic to open for this block. This help topic opens from the **?** button in a separate browser window. See "Add Tooltips and Help in Custom Blocks" on page 409 for more information and examples.
		
		Note: This attribute is only applicable for <presto:help> within <presto:block>.
		
		There are two possible forms to use for relative URLs to help topics:
		
		- *Relative to the MashZone NextGen Server = must* begin with /. There are two ways to host these HTML pages:
		  - `/presto/files/`*some/path/myHelp.html* for help files that have been uploaded to MashZone NextGen. This is a best practice.
		
		  Your MashZone NextGen administrator can host these HTML files in MashZone NextGen by uploading them in the Admin Console.

Name	Required	Description
		Use the URL shown above with the path and file name assigned by your MashZone NextGen administrator. ■ */some/path/in/app-server/myHelp.html* for help files that are hosted in the same application server as the MashZone NextGen Server. ■ *Relative to MashZone NextGenHelp* = `some/path/to/presto-help.html` Relative paths that do not begin with / are *reserved* for MashZone NextGen online help for the built-in blocks in Wires.
key		Reserved for future use.

<icon>

An image to use for this custom block in either the Wires Blocks menu or the canvas. If omitted, Wires uses a default icon for custom blocks. See for details.

If you provide icons, you *must* provide separate images for the menu and the canvas. See "Block Icons" on page 407 for more information on size and visual requirements for block icons.

Can Contain	Empty.
Allowed In	<block>

Attributes

Name	Required	Description
url	yes	The absolute URL to the image for this icon. You can use relative URLs if you or a MashZone NextGen administrator has uploaded the icon to MashZone NextGen.
usage	yes	Where this icon should appear in Wires. ■ `menu` = use this image as the custom block icon in the Wires Blocks menu. ■ `canvas` = use this image as the custom block icon in the Wires canvas.

<uiconfig>

A JSON configuration object for either:

- The entire Block Properties form for this macro.
- The entire configuration for one block property field for this macro.
- Additional configuration along with configuration defined in <block> or in <parameter> for this macro.

Important: When <uiconfig> is used within <block>, you *must* meet the licensing terms for Ext JS as this use is not covered by your MashZone NextGen license.

Familiarity with the Ext JS Library and JavaScript programming in general are required to use <presto:uiconfig>.

Can Contain	Text specifying a configuration object in JSON format that affects either the custom block at the block level or an individual custom block property based on where <presto:uiconfig> is used. If you use this in <presto:block>, the contents are a JSON configuration object for `Ext.form.FormPanel` plus MashZone NextGen-specific properties. This metadata will be used to: - Extend configuration defined in <presto:block>. - Define the entire block properties form, when this element is defined inside <presto:block>. If you use this in <presto:parameter>, the contents are a JSON configuration object for `Ext.form.Field` plus MashZone NextGen-specific properties. This metadata will be used to: - Extend configuration defined in <presto:parameter>. - Define the entire form field corresponding to one input parameter for this macro. See "Advanced Custom Block and Property Configuration" on page 435 for more information. **Tip:** It is a best practice to enclose the JSON configuration object in a CDATA section to ensure that there are no conflicts with the JSON and XML syntaxes.
Allowed In	<block> | <parameter>

<parameters>

Contains custom block configuration for some or all of the input parameters for this macro. These appear as properties in the Block Properties pane in Wires.

Can Contain	<parameter>+
Allowed In	<presto:macro-meta>

<parameter>

Custom block configuration for one input parameter for this macro. Each input parameter for a macro used as a custom block in Wires corresponds to a property for that block in the Block Properties pane. This configuration allows you to have greater control over the visual representation and behavior for a property.

Can Contain	<help>?, <label>?, <required>? <type>?, <uiconfig>?
Allowed In	<parameters>

Attributes

Name	Required	Description
name	yes	The name of the input parameter for this macro that this element defines configuration for.

<required>

A flag indicating whether the block property for this input parameter is required. Wires will not allow users to run a block until all required properties have been set.

Can Contain	`true` or `false`. This defaults to false.
Allowed In	<parameter>

<type>

Additional configuration to define the look or behavior of the block property for this input parameter. See "Configure Strings, Numbers, Dates or Boolean Properties in Custom Blocks" on page 415 for links to more information.

Note: You may specify either <presto:type> or <uiconfig> or both for <presto:parameter>. If you omit both, Wires uses a default input field

combined with the **Path Selection** button for this property. Users can enter a literal value in this field or use the Path Selection button to assign the value dynamically. See "Default Custom Block With No Configuration" on page 404 for an example.

Can Contain	<xpath>?, (<datetimefield> | <list>)?
Allowed In	<parameter>

Attributes

Name	Required	Description
datatype	yes	Additional information that extends the base datatype for this input parameter. ■ `boolean` = renders a selection list with `Yes` and `No` values. Returns `true` or `false` respectively. ■ `datetime` = renders a calendar/time control to easily select dates or times. Use <datetimefield> for additional configuration. ■ `document` = renders an input field but limits users to select results from another block using the **Path Selection** list. Use only when this input parameter is also a `document` type. ■ `enum` = renders a selection list with a set of valid values for this property. You determine the list of valid values using <list> or <uiconfig>. See "Limit Custom Block User Entries to a List of Values" on page 429 for more information. ■ `decimal` = renders a single-line input field that only accepts decimal numbers. ■ `integer` = renders a single-line input field that only accepts integer numbers. ■ `number` = renders a single-line input field that only accepts numbers. ■ `password` = renders a single-line input field that obscures user entries. ■ `path` = renders a single-line input field that only allows users to select a node with the **Path Selection** list. Returns the XPath expression for the selected node as a string. You can limit

Name	Required	Description
		the specific results or path that users see with <xpath>. ■ `string` = renders a single-line input field that accepts text of any kind. ■ `textarea` = renders a multi-line input field. ■ `url` = renders a single-line input field that only accepts string beginning with valid URL protocols.

<datetimefield>

Additional configuration for <presto:type> when `datatype = "datetime"` or when `datatype = "string"`.

Can Contain	Empty
Allowed In	<type>

Attributes

Name	Required	Description
format	yes	Use only with input parameters when `datatype = "string"`. With string parameters, this defines he date and optional time format to apply to the date and time that users choose for this blcok. See "Pick and Format Dates or Times" on page 419 for examples and valid formats.

<list>

Additional configuration for <presto:type> when `datatype` is `enum`.

Use this to configure the list of values that users can select from for a this property. Values for options may be literal or dynamic.

For literal valid values, you can define them as a simple, comma-separated list in <presto:values>. Or, you can use <presto:option> children to provide both the labels that users see and the actual values that are sent to the macro.

For dynamic valid values, you can use <xpath> within <presto:list>. Or you can define dynamic lists using <uiconfig> rather than <presto:type> and <presto:list>. See "Limit

Custom Block User Entries to a List of Values" on page 429 for more information and examples.

Can Contain	(<values> | <option>+ | <xpath>)
Allowed In	<type>

Attributes

Name	Required	Description
usage		Reserved for future use. Currently this renders a selection list combined with the **Path Selection** list button.

<values>

A list of literal valid values for this property.

Can Contain	Text that is valid in XML with values separated by commas.
Allowed In	<list>

Attributes

Name	Required	Description
key		Reserved for future use.

<option>

One literal valid value for this list, with both a label and a value.

Can Contain	Text that is valid in XML that is one value for this list.
Allowed In	<list>

Attributes

Name	Required	Description
label		The label to display in the selection list for this value.
key		Reserved for future use.

<xpath>

The effect of this metadata varies based on where it is used:

- When used inside <presto:type>, this is optional configuration that defines the behavior of the **Path Selection** list for this property. By default, users can assign any block property dynamically using the **Path Selection** list.

  You may use this metadata element to prevent users from entering literal values for a property and limit the choices available in the **Path Selection** list to a single document or a specific branch of nodes within that document. See "Limit Dynamic Choices and Forbid Literal Entries" on page 424 for details.
- When `type="path"` in <presto:type>, you may also use this metadata element to limit the types of nodes that user can choose in the **Path Selection** list. In this case, the block property is assigned the XPath expression for the selected node rather than the actual value of the selected node. See "Get the Chosen Path as Text" on page 426 for details.
- When used inside <presto:list>, this metadata defines the path to obtain a dynamic set of valid values for this list. See "Dynamic List Values from a Macro Input Parameter" on page 432 for details.

Can Contain	Empty
Allowed In	<type> | <list>

Attributes

Name	Required	Description
usage		Wires automatically limits user choices from block results in the **Path Selection** list based on the datatype of the macro input parameter. For input parameters with simple datatypes, for example, Wires only allows users to select simple nodes, also know as *leaf nodes*.

Name	Required	Description
		When <presto:xpath> is a child of <presto:type datatype="path">, automatic detection is not possible. This attribute allows you to control the types of nodes that users may select in that case: ■ `array` = users may select only repeating nodes, simple (with data) or complex (with children) ■ `branch` = users may select only complex nodes with children, either a single node or repeating nodes ■ `leaf` = users may select only a single simple node with data or a repeating simple node ■ `path` = users may select any type of node ■ If omitted = users may select any type of node
limitTo	Yes	Limits the choices presented to users in the **Path Selection** list to nodes in a specific document-type input parameter, and optionally nodes within a specific path in that document. When <presto:xpath> is defined as a child of <presto:list>, this attribute identifies a document-type input parameter, and optional path within that document, that provides the valid values for the list.

<presto:presto-meta>

AMashZone NextGenextension statement for built-in MashZone NextGen metadata properties to assign to mashups.MashZone NextGendefines and uses built-in metadata properties to provide additional information forMashZone NextGencomposers.

Important: Effective with this release, the uses and variations for <presto:presto-meta> in macros are *deprecated*. Instead, use <presto:macro-meta>. See "created-using" on page 594 for the remaining use of <presto:presto-meta>.

Can Contain	Can contain text and any well-formed literal XML. Valid content depends on the value of the `name` attribute.
Allowed In	mashup | macro | operation

Attributes

Name	Required	Description
name	yes	`created-using` = identifies the MashZone NextGen composer that was used to create this mashup. This flag is used to distinguish which mashups can open in Wires.

created-using

Indicates the MashZone NextGen composer used to create a mashup script. Valid values include `Wires` or blank. Used in Wires to determine which mashups are editable. For example:

```
<presto:presto-meta name="created-using">Wires</presto:presto-meta>
```

<user-meta>

One user-defined metadata property to assign to this mashup. User-defined property names should be unique within a given mashup script and cannot use reserved property names. You assign the value for this property as the value of this element.

See also "Adding Metadata to Mashups" on page 581 for more information and examples.

Can Contain	Can contain text and any well-formed literal XML. Valid content is user defined.
Allowed In	mashup | macro | operation

Attributes

Name	Required	Description
name	yes	The name of this user-defined metadata property.

Migrating Mashups to a New EMML Version

You may need to migrate mashup scripts for new versions of EMML if:

- You wish to use new features or syntax from the new EMML schema.

- Your existing mashup script uses a feature or namespace that is no longer supported. See release notes for information on EMML feature support.

Migrating Mashups to Use a New Feature or Namespace

To use a new feature from the EMML schema in an existing mashup script, you must update the EMML namespace in your mashup script. You may also need to change the schema location.

For the Mashup Editor, the new schema is installed with the new version of MashZone NextGen.

1. Open theEMMLfile in theMashup Editor.
2. In the <mashup> element, change the namespace in the following attributes to match the new EMMLversion:
   - *xmlns* = the default namespace
   - *xsi:schemaLocation* = the location of the EMML schema

   See "EMML Namespaces" on page 450 for a list of supported namespaces. With an original namespace like this:

```
<?xml version="1.0"?>
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.jackbe.com/2008-03-01/EMMLSchema"
  xsi:schemaLocation="http://www.jackbe.com/2008-03-01/EMMLSchema
     ../xsd/EMMLSpec.xsd"
  name="myMashup">
```

   You would change the namespace to something like this:

```
<?xml version="1.0"?>
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
     ../xsd/EMMLSpec.xsd"
  name="myMashup">
```

3. If the schema file name or location has changed, change the path information in the *xsi:schemaLocation*attribute.

   With an original `xsi:schemaLocation` like this:

```
<?xml version="1.0"?>
<mashup
  xmlns="http://www.jackbe.com/2007-09-15/JMMLSchema"
  xsi:schemaLocation="http://www.jackbe.com/2007-09-15/JMMLSchema
     ../xsd/JMMLSpec.xsd"
```

   You would change it to something like this:

```
<?xml version="1.0"?>
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
     ../xsd/EMMLSpec.xsd"
```

4. If the mashup uses macros or you want to add macros, you must add or update the *xmlns:macro*attribute.

   You would add this attribute or change it to something like this:

```
<?xml version="1.0"?>
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
     ../xsd/EMMLSpec.xsd"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  name="myMashup">
```

5. If the mashup or macros use MashZone NextGenextension elements, such as <presto:presto-meta>, you must add or update the *xmlns:presto*attribute.

   For example:

```
<?xml version="1.0"?>
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
     ../xsd/EMMLSpec.xsd"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/schemas/v1.0/EMMLPrestoExtensions"
  name="myMashup">
```

6. Edit the mashup script, if needed, and save your changes.

7. Test the new mashup script. See"Test Mashup Scripts or Macros in Mashup Editor" on page 683or"Testing Mashup Scripts from the Command Line" on page 665for instructions.

Expressions for Mashups

You may find A Brief Introduction to XPath 2.0 helpful as a starting point for working with XPath expressions in mashups. You can also create your own custom XPath functions to use in expressions. See "Defining Custom XPath Functions" on page 601 for information.

In addition, expressions can be dynamic in mashups. You can use Dynamic Mashup Expressions directly in XPath. Or, use Dynamic XPath or Other Syntax in Mashups to pass dynamic portions of XPath as parameters. Finally, see "Using XQuery Expressions in Construction Statements" on page 603 for information on dynamic expressions that you use to help construct the mashup result or intermediate variables.

A Brief Introduction to XPath 2.0

The MashZone NextGen Server uses XPath 2.0 expressions to work with the results of component mashable information sources and all the variables used for a mashup script. This topic introduces the simplest XPath capabilities that you can use in a mashup script. It uses the shortcut XPath syntax.

Note: Readers should be familiar with basic XML terminology.

XPath Nodes

XPath receives an XML document from a parser as a tree of nodes. XPath 2.0 *nodes* are the:

- XML document and its:
  - Elements
  - Attributes
  - Processing instructions
  - Comments
  - Text
  - Namespaces

Paths

The most basic expression is a *path* that traverses the document nodes to a specific node or a set of nodes called a *sequence* or *node set*. For example:

```
/document/chapter/section/paragraph
```

This path points to a <paragraph> element inside <section> inside <chapter> inside <document> which is the root node for the XML document. The slashes (/) separate each *step* in the path from parent to child.

There are several important points to note about this example:

- **Traversal Order**: the path traverses the XML document in document order because this is the default direction for paths. XPath expressions can traverse the document in reverse order or skip levels as needed.
- **Absolute Path**: this example is an absolute path because it starts with a slash (/). This initial slash is the root node in the expression - a starting point just before the document node that encloses all of the document.

  If you omit the initial slash, the path is relative to the current position in the document - called the *current context*. Most mashup expressions use absolute paths or they are relative to a context that is defined in the mashup script.
- **Hierarchy**: this example is explicit about the whole hierarchy to traverse. It would not, for example, match any <paragraph> nodes inside <section> that was inside another <section>.

  You can use a double slash (//) to indicate that the expression can skip any number of levels within the tree. You can use this at the beginning of absolute paths or at any step within a path. The two examples shown below would find any <paragraph> node at any level:

  ```
  /document//paragraph
  //paragraph
  ```

- **Cardinality**: this example more than likely results in a set of nodes rather than one specific <paragraph> element. The path matches any <paragraph> in any <section> in any <chapter> in <document>. If no <chapter> contains a <section>, this path would result in an empty sequence with no nodes.
- **Elements Only**: this path matches <paragraph> elements only. Any attributes or descendants of <paragraph> are included, but no other types of nodes are selected.

To match attributes, add a step in the path after the element the attribute belongs to and use "@" in front of the attribute name. For example:

```
/document/chapter/section/@id
```

This matches the id attribute on any <section> element in any <chapter> in <document>.

Namespaces and Node Names

Namespaces appear as *prefix* : before a node name in some XML documents. For example: `xs:para` has a namespace *xs*. Namespaces can be quite confusing, as the prefix is simply a shorthand for the actual namespace that is identified with an `xmlns` attribute. Frequently, the full namespace is declared on an ancestor, such as `xmlns:xs="http://myNamespace"`.

Nodes with namespaces belong to a specific category. Nodes with the same name but a different namespace - or no namespace - **are not the same type of nodes**. The namespace modifies the node name into a different group.

The two path expressions shown below select different `title` nodes:

```
/document/chapter/section/a:title
/document/chapter/section/title
```

The first path selects any section `title` node in the namespace associated with the **a** prefix. The second path selects any section `title` node that has *no* namespace.

You can use wildcards in XPath expressions to ignore namespaces or to ignore node names:

- ***:***node-name* in an XPath expression selects all nodes with the matching name, regardless of what namespace they belong to, including having no namespace.
- *namespace* **:*** in an XPath expression selects all nodes belonging to a specific namespace, regardless of the node name.

Using Predicates to Select or Filter

You can use predicates in XPath expressions to make the match more specific or to filter out nodes. Predicates appear within the steps of a path and inside brackets, such as this example:

```
/document/chapter/section[@id='intro']
```

The predicate `[@id='intro']` makes this expression match only <section> element with an id attribute value of `intro`. You can use predicates to test many types of conditions:

- **Existence**: `/document/chapter/section[@id]` matches any <section> that has an id attribute.
- **Position**: `/document/chapter/section[3]` matches the third <section> of any <chapter>. `/document/chapter/section[last()]` matches the last <section> of any <chapter>.
- **Combining Criteria**: you can use the keywords `and` or `or` to combine the criteria in a predicate. You can also specify multiple predicates. They are evaluated in order.

  The example shown below selects all <section> nodes that belong to a <chapter> with a <role> child and a position of 3 or greater in <document>.

  ```
  /document/chapter[role and (position() > 2)]/section
  ```

  This expression could also be done with:

  ```
  /document/chapter[role][position() > 2]/section
  ```

- **Arithmetic and Logical Operators**: predicates can use common logical operators such as = or >. You can also perform basic arithmetic such as (2 + 3) in predicates.

  Note: You may need to use XML escaping with the arithmetic opertors that use the < or > characters. See "XML Escaping in URLs and Expressions" on page 600 for information.

- **Comparison Functions**: predicates can also contain XPath functions that affect the comparison or define a specific relationship. Common examples include:

  - `not()` to negate the expression. This example selects any <section> child of <chapter> that does not have an id attribute set:

    ```
    //chapter/section[not(@id)]
    ```

  - `contains()` to determine string inclusion. This uses case-sensitive comparisons. This example selects any <item> node that has a <category> child whose value contains `ASST` somewhere in the string:

    ```
    //item[contains(category,'ASST')]
    ```

  - `matches()` which also determines string inclusion using regular expressions. You can also do case-insensitive comparisons. This is a case-insensitive comparison, for example:

    ```
    //item[matches(category,'ASST','i')]
    ```

Referencing Variables

You can refer to variables in XPath expressions using `$`*variable-name* . Variables can also occur at the beginning of paths, in calculations, or in predicates.

Performing Arithmetic with XPath

Simple arithmetic expressions are also valid XPath expressions. You can use any of the following expressions as XPath:

```
1 + 1
/order/item/price - 1.00
```

```
/order/subtotal * /order/taxrate
($total div 100) + 10
$total idiv 100
/order/qty mod 10
```

XPath uses `div` for decimal division and `idiv` for integer division. Similarly, `mod` is the modulo operator. You can also use parentheses to group arithmetic expressions and control precedence.

As these examples show, you can combine paths and arithmetic in expressions to perform calculations. The value of the path must result in a numeric value, or a string value that lexically represents a number, for the calculation to be successful. If the path cannot be resolved to a number, or the calculation is invalid (such as division by zero), the result is `NaN` (not a number).

XPath Functions to Transform Node Data

You can also use any XPath 2.0 function in any step of a path or expression. Functions allow you to express things such a last() or not(). The following example selects any <chapter> nodes that do not contain a <role> child:

```
/document/chapter[not(role)]
```

In addition, there are many string, numeric and date functions that let you transform the text in elements or attributes. This example changes all the text of the first <paragraph> nodes to upper-case:

```
upper-case(//paragraph[1])
```

You can also use XPath functions to cast node data to other data types. This example casts the value of the first <highQuote> node to decimal.

```
xs:decimal(//stock[1]/highQuote)
```

XPath functions also let your retrieve the current date or date and time or perform date calculations. This example calculates the number of days between a <comment> posted date and the current date.

```
op:subtract-dates(xs:date(//comment/@posted),fn:current-date())
```

For More Information

This introduction touches only the most basic and simple aspects of XPath 2.0. For more information about syntax and the available functions, see the XPath 2.0, XPath 2.0 functions and the XPath 2.0 Data Model specifications.

See also "XML Escaping in URLs and Expressions" on page 600.

XML Escaping in URLs and Expressions

XPath expressions, URLs, SQL expressions or other values appear in EMML statements or in the XML requests built in the Service Inspector view. All of these expressions can contain characters that are XML delimiters and thus can cause mashups to fail or cause problems testing services in Service Inspector.

As a consequence, the following XML delimiter characters *must* be escaped in EMML or in input parameters in Service Inspector:

Character	XML Escape
&	&
<	<
>	>
"	"
'	'

You only need to escape quotation characters (single or double) within attributes. For example:

```
<filter inputvariable="queryResult"
  filterexpr="/customers[capitalization &gt; 125]"
  outputvariable="midRange"/>
<directinvoke
  method="GET"
  outputvariable="result"
  endpoint="http://www.mySearch.com?q='java'&amp;v='2.1'" />
```

Defining Custom XPath Functions

You can create custom XPath functions to perform calculations and use these functions in XPath expressions anywhere in mashups or macros.

To define custom functions:

1. Write a Custom XPath Function Class
2. Add the Function to the Classpath for the MashZone NextGen Server
3. Add the Function to a Mashup Script or Macro

Write a Custom XPath Function Class

You create the function as a Java class that extends `com.jackbe.presto.mashup.EMMLUserFunction`. See MashZone NextGen Custom XPath API for reference information on this class.

Note: In earlier MashZone NextGen releases, this API was the `com.jackbe.jbp.jems.client.EMMLUserFunction` class. Any method in your custom class can be used as an XPath function with these limitations:

- The method must be a static class method.
- All parameters for the method must be simple types.
- The return type can be `void` or any simple type. It cannot return a node or node set.

For example:

```
package com.mycompany.mashups;
import com.jackbe.presto.mashup.EMMLUserFunction;
public class CustomFunctions extends EMMLUserFunction {
  public static boolean soundex(String name, String pattern){
    org.apache.commons.codec.language.Soundex s = new;
    return s.encode(name).equalsIgnoreCase(s.encode(pattern));
  }
}
```

You must add the following JARs to the classpath to compile your custom XPath Function class:

- *web-apps-home* /presto/WEB-INF/lib/emml.jar
- *web-apps-home* /presto/WEB-INF/lib/presto-api.jar
- *web-apps-home* /presto/WEB-INF/lib/saxon8.jar

Add the Function to the Classpath for the MashZone NextGen Server

You must add custom XPath function classes to the MashZone NextGen Server for any mashup scripts that uses this function.

To add the function to the classpath for the MashZone NextGen Server, you copy either the compiled class file or a JAR containing the compiled class file for the function to one of these folders, respectively:

- The external configuration folder, if any, for the MashZone NextGen Server.

  Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

- *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
- *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

Add the Function to a Mashup Script or Macro

Once you have added the function to the classpath, you can use it in mashup scripts or macros. First, you must declare a namespace for the custom function class using the `java:` protocol and the fully qualified Java class name. For example:

```
<?xml version="1.0"?>
<mashup name="SoundexSearch"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
```

```
    ../schema/EMMLPrestoSpec.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:myFunc="java:com.mycompany.mashups.CustomFunctions" >
...
```

Then use the function, with its namespace, in any XPath expression in the mashup script. This example uses the function in a filter expression:

```
...
  <input name="param" type="document">
    <data>
      <item><name>test1</name></item>
      <item><name>data2</name></item>
      <item><name>test3</name></item>
      <item><name>data4</name></item>
      <item><name>data-5</name></item>
      <item><name>data.6</name></item>
      <item><name>data-6-6</name></item>
      <item><name>5-6-data-6-6</name></item>
    </data>
  </input>
  <output name="result" type="document"/>
  <filter inputvariable="param"
      filterexpr = "/data/item[myfunc:soundex(name, 'data')]"
      outputvariable="result"/>
```

Notice that the first parameter for the function is passed with a relative XPath expression to the name node. You can use XPath expressions as function parameters as long as the expressions resolve to a single node with simple content.

Using XQuery Expressions in Construction Statements

XQuery is another query syntax for XML, like XPath, that uses a syntax similar to SQL. XPath expressions are also valid XQuery expressions, but XQuery includes some additional features to handle more complex queries.

You can use XQuery expressions as Dynamic Mashup Expressions inside literal XML for the <constructor> and <appendresult> statements. All of the variables and input parameters for the mashup script are available within XQuery, allowing you to manipulate the results from mashable information sources.

Some tasks that XQuery is useful for include queries that iterate through several mashables results, queries that should be sorted before constructing results or queries that require complicated calculations. For more information, try the simple XQuery tutorial at http://www.stylusstudio.com/xquery_primer.html or see the XQuery Specification.

FLWOR Syntax

The basic syntax for XQuery is known as *FLWOR*:

- *for*: to iterate through variable data.
- *let*: to define new variables, either constructed as XML or using functions to other XQuery or XPath expressions to calculate values.
- *where*: to specify the criteria used to select data. This can also act like a join.

- *order*: to sort the selected, calculated and joined results.
- *return*: to construct the actual XML to return.

The example shown here illustrates the basic syntax of all for the FLWOR statements:

```
<constructor outputvariable="$result" >
  <res:result>
    {
      for $i in distinct-values($demographics//zip-code),
          $j in distinct-values($weblogs//zip-code)
      let $purchases := $weblogs/weblog/record[ zip-code = $j],
          $total-purchases := sum(data($purchases//purchase-amount)),
          $population := $demographics/demographics/person[zip-code = $i],
          $income-avg := avg(data($population//income)),
      where $i = $j
      order by $total-purchases descending
      return <res:item
                zip-code="{$i}"
                total-purchase="{$total-purchases}"
                income-average="{$income-avg}" />
    }
  </res:result>
</constructor>
```

Dynamic Mashup Expressions

Dynamic mashup expressions provide dynamic content in a mashup script. You can use dynamic mashup expressions:

- **In literal XML**: to define the data to fill the mashup result, variable or parameter that is being constructed. See "Dynamic Expressions in Literal XML" on page 605 for examples
- **In most XPath expressions or EMMLattributes that accept an XPath expression**: This can replace the entire XPath expression or any part of the XPath expression.

  Note: You cannot nest dynamic mashup expressions. So you cannot use a dynamic mashup expression in an XPath inside a dynamic mashup expression for literal XML. This can be useful to make any EMML statement dynamic including macros. See "Dynamic Expressions in XPath Expressions" on page 605 and "<macro>" on page 638 for examples.

- **In <template> elements to build syntax or any string dynamically**: This is commonly done to build a dynamic expression to use as input parameters for a generic mashup script or generic macro. But you can use the <template> element to build any string dynamically within a mashup script or macro. See "Dynamic XPath or Other Syntax in Mashups" on page 606 for examples.
- **In <constructor> or <appendresult>**: with XQuery expressions. See "Using XQuery Expressions in Construction Statements" on page 603 for examples.

Dynamic expressions are XPath expressions enclosed in braces ({ *expression* }). Variable references start with $ in dynamic mashup expressions. For example:

```
{$filterResult/list/name}
```

```
{upper-case($list/memberName)}
{$selectedItem}[matches(description, 'Ruby')
Hello, my name is {$currentUser}!]
```

Dynamic Expressions in Literal XML

Variable or parameter declarations and mashup statements that define the structure of the result returned by that statement can include the literal XML structure of the content and literal data. These statements can also use dynamic mashup expressions to define the data that should fill this literal structure dynamically.

For example:

```
<constructor outputvariable="$result">
    <res:mailingList>
      <res:displayName>{$filterResult/list/name}</res:displayName>
      <res:email>{$filterResult/list/email}</res:email>
      <res:greet>Hello {$filterResult/list/name}, </res:greet>
    </res:mailinglist>
</constructor>
```

Mashup expressions can also contain XPath functions or predicates, such as this example:

```
<appendresult outputVariable="$result">
  <res:member>
    <res:name>{upper-case($list/memberName)}</res:name>
    ...
  </res:member>
</appendresult>
```

These dynamic expression can use XQuery expressions instead of XPath for more complicated construction scenarios such as iterating through several variables or using sorting. See "Using XQuery Expressions in Construction Statements" on page 603.

Dynamic Expressions in XPath Expressions

XPath expressions used in a mashup script or macro can contain dynamic mashup expressions. This allows the behavior of the mashup script or macro to be dynamic based on input parameters.

In the example shown below, the path that defines the nodes to use in a <filter> statement is defined in a variable named `selectedItem`. The variable must be resolved first, to get the full XPath expression to use for filtering.

```
<filter outputvariable="$result"
  filterexpr="{$selectedItem}[matches(title, $feedfilter)]"/>
```

This next example also shows a filter where only a portion of the XPath that selects nodes is from a dynamic mashup expression:

```
<filter outputvariable="$result"
  filterexpr="$documents/product/{$version}/]"/>
```

Dynamic expressions can also be passed from a <template> declaration. For example:

```
<template expr="{$selectedItem}[matches(title, $feedfilter)]"
  outputvariable="inputFilter"/>
<filter outputvariable="$result" filterexpr="$inputFilter"/>
```

Dynamic XPath or Other Syntax in Mashups

You can use the <template> declaration to create XPath expressions, SQL queries or other syntax strings dynamically for any use in a mashup or a macro. This may define and use dynamic expressions in a single mashup or macro. Or it can be used to pass dynamic input to another, generic mashup script or generic macro.

You define the syntax string you want with one or more dynamic mashup expressions in the `expr` attribute for <template>. This expression is evaluated at runtime and the result is assigned to the variable you identify in the `outputvariable` attribute.

The output variable can then be used in other EMML statements or passed as an input parameter to other mashups or macros.

Examples

The example shown below defines an expression for a join condition. In this case, `$column1` and `$column2` are input parameters that contain the XPath expressions to the columns to use in the join. This allows the subsequent <join> statement to be dynamic based on user inputs.

```
<template expr="{$column1} = {$column2}" outputvariable="$dynamicEqualJoin"/>
...
<join outputvariable="$joinResult" joincondition="$dynamicEqualJoin"/>
```

This example shows <template> used to construct a SQL query:

```
<input name="offense" type="string" />
<input name="method" type="string" />
<input name="numrows" type="number" default="10" />
...
<variable name="sqlquery" type="string"
  default="select * from crime where 0=0  " />
...
<if condition="$offense != ''">
    <template expr="{$sqlquery} and offense = '{$offense}' "
      outputvariable="sqlquery"/>
  <elseif condition="$method != ''">
    <template expr="{$sqlquery} and method = '{$method}' "
      outputvariable="sqlquery"/>
  </elseif>
</if>
<template expr="{$sqlquery} limit 0,{$numrows}" outputvariable="sqlquery" />
...
<sql name="DCCrime2008" query="$sqlquery" outputvariable="result"/>
```

Advanced Mashup Techniques

Mashups can be quite complex. Some of the intermediate or advanced techniques that you may find useful include:

- Adding HTTP Headers to the Mashup Result
- Adding Authentication Headers for Component Mashables

- Normalizing Data for Joins, Grouping or Filtering
- Adding Special Characters to Data
- Converting Numbers in Floating Point Notation
- Using MashZone NextGen Attributes in Mashups to have mashable or mashup parameters supplied dynamically by the MashZone NextGen Server.
- Removing Duplicates With Filtering
- Setting the Mashup Response Character Encoding
- Wrapping Results or Variables in CDATA Sections
- Web Clipping with a Mashup Script to use any well-formed HTML response as data that you can manipulate with a mashup.
- Handling HTML Responses
- Handling JSON Responses or Inputs
- Wrapping POJO Classes with Mashups to publish Java classes as MashZone NextGen mashables or simply to use Java classes within the processing flow of a mashup.
- Wrapping WSDL Web Services That Return HTML Results
- Pre-Processing/Post-Processing Mashables with Mashups
- Using XQuery for Outer Joins
- Defining and Using Custom Mashup Statements with Macros
- Dynamic Mashup Syntax to create mashup scripts that can be applied to many mashables or have dynamic expressions that are passed in.
- Generating Mashup Scripts Dynamically to allow the logic of a mashup to change dynamically.
- Using the Built-in Mashup Query Web Service to retrieve metadata about mashups.

Adding HTTP Headers to the Mashup Result

You can define HTTP headers to send with a mashup result when the mashup is invoked in MashZone NextGen. For any HTTP header, you add <variable> declarations using reserved names in the form `httpResponse.`*header-name*.

The *header-name* portion of the variable name can be any standard HTTP header or a custom header. Custom header names must begin with `X-`. For example:

```
<variables>
  <!-- standard content type HTTP header for a result is CSV format -->
  <variable name="httpResponse.Content-Type" type="string" default="text/csv"/>
  <!-- standard last modified HTTP header -->
  <variable name="httpResponse.Last-Modified" type="datetime"/>
  <!-- custom HTTP header -->
  <variable name="httpResponse.X-CUSTOM-HEADER" type="string"/>
```

```
</variables>
<assign outputvariable="$httpResponse.X-CUSTOM-HEADER" fromexpr="....."/>
```

Adding Authentication Headers for Component Mashables

Some mashable information sources require authentication information when they are invoked. Typically, you publish these mashables in MashZone NextGen and provide default authentication credentials.

You can override the default credentials when you invoke these mashables using optional MashZone NextGen headers. See "MashZone NextGen Headers/Parameters" on page 1511 for more information.

You can also use these optional headers in mashup scripts. You must Construct Mashable Authentication Headers and then Invoke the Mashable with Authentication Headers. See also "Example 17. Adding HTTP Basic Authentication to <directinvoke> Requests" on page 488.

Construct Mashable Authentication Headers

Use <constructor> or <variable> with literal XML to create the optional MashZone NextGen header for authentication credentials for the component mashable in your mashup script. The name of the header becomes the root element within the variable and each property is a child element.

Basic HTTP authentication uses the `httpBasicAuth`MashZone NextGen header with `username` and `password` properties. For example:

```
<variable name="basicAuthHeader" type="document">
  <httpBasicAuth>
    <username>svcuser</username>
    <password>svcpw</password>
  </httpBasicAuth>
</variable>
```

Authentication for Windows NT Domains uses the `ntlmAuth` and adds a `domain` property:

```
<variable name="ntlmAuthHeader" type="document">
  <ntlmAuth>
    <username>svcuser</username>
    <password>svcpw</password>
    <domain>ntDomain</domain>
  </ntlmAuth>
</variable>
```

Invoke the Mashable with Authentication Headers

Once you have constructed the headers simply invoke the service and pass them with the `header` attribute in <invoke>:

```
<invoke service="someService" operation="opA" header="$basicAuthHeader" output="result"/>
```

Normalizing Data for Joins, Grouping or Filtering

In many cases, the results from different mashables may contain similar data that is not expressed in identical forms. This makes the comparisons used in joins, grouping or filtering difficult or impossible until you normalize the data to a single representation.

When you need to normalize data for comparisons, the best method is to create a custom XPath function that you add to the MashZone NextGen Server. This allows you to:

- Use the custom function directly in the XPath expression that is comparing the disparate data. Thus the comparison is handled properly, but mashable results are not altered.
- Reuse data normalization logic in any mashup that you publish to the MashZone NextGen Server where the custom function is deployed.

> **Note:** If the data normalization logic is specific to one mashup and you have no need to reuse this, you can also do data normalization using scripting and the <script> statement.

See "Defining Custom XPath Functions" on page 601 for complete instructions on how to write custom XPath functions and deploy them in the MashZone NextGen Server. Once you have your custom function deployed, you simply declare a namespace for the function in any mashup script or macro and use the function in the appropriate mashup statement.

Example Custom XPath Function for Data Normalization

This example shows a very simple data normalization function and the mashup script that uses the custom function to join the results of two mashables. The example joins mortgage rate information from two web sites. One site refers to the APR, but the second uses custom terms specific to their mortgage products to refer to rates. To combine the results for comparison, the custom terminology must be normalized.

First, you create the custom XPath function logic in a Java class that extends `org.oma.emml.client.EMMLUserFunction`. This class looks something like this example:

```
package com.mycompany.mashups;
import org.oma.emml.client.EMMLUserFunction
...
public class MyFinanceFunctions extends EMMLUserFunction {
static Set mortgageAliases = new HashSet();
static { mortgageAliases.add("5/1 Orange Mortgage"); }
public static String mortgage(String data) {
  if (mortgageAliases.contains(data))
    return "5-Year ARM";
  return data; }
}
```

Then compile and deploy the class to the MashZone NextGen Server that will host mashups that need to use this function. See "Defining Custom XPath Functions" on page 601 for instructions.

Use the Custom XPath Function in Your Mashup or Macro

To use the function, you must add a *namespace* for the class that contains this function. Add an `xmlns` attribute with a unique namespace prefix to the <mashup> or <macro> tag. For example:

```
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
    ../xsd/EMMLPrestoSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:finance="java:com.mycompany.mashups.MyFinanceFunctions"
    name="MortgageComparisons">
  ...
</mashup>
```

Then simply use the custom function in the XPath expressions where you need data to be normalized. In this example, the custom XPath expression is used in a <join> statement:

```
...
  <input name="feed1" type="document">
    <feed>
      <Product>5-Year ARM</Product>
      <Rate>5.250%</Rate>
      <APR>5.388%</APR>
    </feed>
  </input>
  <input name="feed2" type="document">
    <feed>
      <Product>5/1 Orange Mortgage</Product>
      <Rate>5.500%</Rate>
      <APR>4.872%</APR>
    </feed>
  </input>
  <output name="result" type="document"/>
  <join outputvariable="$result"
    joincondition="$feed1/feed/finance:mortgage(Product) =
    $feed2/feed/finance:mortgage(Product)"/>
  <display message="result = " expr="$result"/>
....
```

Converting and Working with Dates and Times

Mashable or mashup results frequently contain date or time data in various string formats. Sorting, filtering or performing calculations requires converting the string representations of the dates or times to an acceptable format for XPath, which uses XML Schema datatypes. These are based on ISO 8601 formats:

`yyyy-MM-ddThh:mm:ss.s zzzzzz`

For details, see XML Schema Part 2: Datatypes.

In most cases, you can use the MashZone NextGen custom XPath function `ISODateFormatExt` to convert dates or times to a string in the ISO format. Then use an XPath datatype function to cast the string to a date or time.

Note: If your mashup script puts dates or times in variables, use the `datetime` datatype for the variable. The `datetime` datatype provides more reliable behavior than the `date` datatype which has been *deprecated*.

Using the ISODateFormatExt Custom XPath Function

This custom XPath function is built into MashZone NextGen. To use the function in an XPath expression you must:

1. Declare the MashZone NextGen custom XPath namespace on <mashup> in your mashup script. For example:

```
<mashup name="dateExample"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
      ../xsd/EMMLPrestoSpec.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:fn="java:com.jackbe.jbp.jems.client.EMMLPrestoFunctions">
...
</mashup>
```

2. Define the to and from formats for the date or time you need to convert.

   For example:

```
<mashup name="dateExample"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
      ../xsd/EMMLPrestoSpec.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:fn="java:com.jackbe.jbp.jems.client.EMMLPrestoFunctions">
  <output name="result" type="document"/>
  <variable name="fromFormat" type="string" default="dd/MM/yyyy"/>
  <variable name="isoFormat" type="string" default="yyyy-MM-DD"/>
...
</mashup>
```

   These formats use patterns to identify the various date or time components. Some of the most common include:

Pattern	Usage
dd	Day of the month such as 05.
yyyy	Year.
MM	Numerical month.
MMM	Text month
hh	Hour in a 12-hour format (1-12).

Pattern	Usage
HH	Hour in a 24-hour format (0-23).
a	AM/PM for 12-hour time formats.
mm	Minute
ss	Second
S	Millisecond

The complete list of characters and symbols that are valid in patterns are defined in the `java.text.SimpleDateFormat` class. For detailed information, see Java API documentation for the JDK version used in your environment.

3. Use the function in an XPath expression. In most cases you would use this function along with an XPath constructor function to obtain a date or time.

   For example:

```
<mashup name="dateExample"
   xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
   xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
     ../xsd/EMMLPrestoSpec.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:fn="java:com.jackbe.jbp.jems.client.EMMLPrestoFunctions">
 <output name="result" type="document"/>
 <variable name="fromDate" type="string" default="dd/MM/yyyy"/>
 <variable name="isoDate" type="string" default="yyyy-MM-DD"/>
 <invoke service="YahooRSSFeed" operation="GET"
     outputvariable="$stories"/>
 <sort inputvariable="$stories" sortexpr="/rss/channel/item"
   sortkeys="xs:date(fn:ISODateFormatExt(pubdate, $fromFormat, $isoFormat))"
   outputvariable="result" />
</mashup>
```

ISODateFormatExt(*dateText*, *fromFormat*, *toFormat*)

Parses a string representation of a date and/or time in the specified format and returns a string representation of the date and/or time in the ISO 8601 format.

dateText	String	A string or an XPa
fromFormat	String	The date or time pa `java.text.Simpl`
toFormat	String	The date or time pa

Converting Numbers in Floating Point Notation

The results of calculations in EMML using XPath expressions can end up in floating point notation, such as `1.625E10`. To convert floating point numbers back to decimals, simply use the `xs:decimal` XPath function. For example:

```
...
  <assign fromexpr="$total * $factor" outputvariable="$largeNumber"/>
  <if condition="contains(string($largeNumber,'E'))">
    <assign fromexpr="xs:decimal($largeNumber)" outputvariable="$largeNumber"/>
  </if>
...
```

Adding Special Characters to Data

When you transform data in EMML, you may need to add white space, control or other special characters to data. For example, transforming data into a URL that has parameters and uses the `&` character or adding line feed characters in a string.

Because EMML is XML, you can do this using XML character entities for the special characters and XPath functions to append the characters.

The next example shows an <assign> statement that adds a line feed to the end of a string:

```
<assign fromexpr="concat($line,'&#010;')" outputvariable="$line" />
```

The second string, `
`, is the XML character entity for a line feed character.

The next example loops through an XML variable with parameters to add to a URL:

```
...
<!-- If there are parameters, add ? -->
<if condition="$svcResult/parameters/parameter">
  <assign fromexpr="concat($thisUrl, '?')" outputvariable="$thisUrl"/>
</if>
<foreach variable="thisParam" items="$svcResult/parameters/parameter">
   <assign fromexpr="concat($thisURL, $thisParam/name, '=', $thisParam/value)"
     outputvariable="$thisUrl"/>
  <!-- If there are more parameters, add & -->
  <if condition="$thisParam/following-sibling::parameter">
    <assign fromexpr="concat($thisUrl,'&amp;')" outputvariable="$thisURL"/>
  </if>
</foreach>
...
```

XML Character Entities

XML has character entities predefined for all XML delimiters such as < or &. In addition, XML supports character entities for *any* Unicode character in one of two forms:

- `&#`*nnn*`;` = where *nnn* is the decimal number for the Unicode character.
- `&#x`*nnn*`;` = where *nnn* is the hexadecimal number for the Unicode character.

Some of the most common special characters you may want to use include:

Character	Named Form	Decimal Form	Hexadecimal
<	<	<	<
>	>	>	>
&	&	&	$#x26;
non-breaking space			
line-end	
tab			

For a complete list of Unicode character points, see the Unicode Code Charts or the character Name Index.

Using MashZone NextGen Attributes in Mashups

You can use attributes defined for MashZone NextGen to provide input parameters or other variables in mashups. This is also sometimes called *server-side binding* or *server templating*.

You declare variables for MashZone NextGen attributes, just as you would for any other variable. The value for the variable is resolved by the MashZone NextGen Server at run-time from a MashZone NextGen attribute.

MashZone NextGen attributes can be defined for individual users, at a global level, within a user session or for a specifc HTTP header or cookie for a specific request. You can also define *artifact attributes* as extensions for specific mashables or mashups.

Mashup support for MashZone NextGen attributes is:

- Full support for MashZone NextGen user and global attributes. See "MashZone NextGen Global and User Attributes" on page 615 for details.

  For example, some mashable information sources require authentication or application IDs. These may be for a specific user or all users may use a single ID.
- Partial support MashZone NextGen session attributes. See "MashZone NextGen Session Attributes" on page 615 for details.
- No support for request header or cookie attributes.
- No support for artifact attributes.

See also "Using MashZone NextGen Attributes in Mashup Statements" on page 616 for examples.

MashZone NextGen Global and User Attributes

MashZone NextGen user and global attributes use the following reserved variable names:

- *user*: for MashZone NextGen user attributes. Users can add custom attributes to their MashZone NextGen profile. See "Manage Your MashZone NextGen User Attributes" on page 82 for more information.

  Attributes for users from the User Repository can also be exposed as MashZone NextGen user attributes. For more information on which user attributes are accessible from the User Repository.

  - Declare a MashZone NextGen user attribute in the mashup script as:

    `<variable name="user.`*attribute-name*`" type="string"/>`

  - Once declared, use a MashZone NextGen user attribute as:

    `$user.`*attribute-name*

- *global*: for MashZone NextGen global attributes. Global attributes are defined in the Admin Console. .

  Use this syntax:

  - Declare a MashZone NextGen global attribute in the mashup script as:

    `<variable name="global.`*attribute-name*`" type="string"/>`

    Note: Earlier releases also used "system" as a keyword for global attributes. This keyword is deprecated, but still supported.

  - Once declared, use a MashZone NextGen global attribute as:

    `$global.`*attribute-name*

Important: You must *always* use the `$`*variable-name* syntax for MashZone NextGen attributes, *even* for input variables in the <invoke> statement.

MashZone NextGen Session Attributes

MashZone NextGen session attributes use the reserved variable name *session*. So you declare session attributes as `session.`*attribute-name*, just as you would with any variable.

The values for session attributes, however, can also be *set* in two ways:

- Using the `x-p-sessionbinding`MashZone NextGen Header/Parameter in requests to the MashZone NextGen Server to save specific data from a mashable or mashup response as a session attribute.

You can send this as an HTTP header or as a parameter using MashZone NextGen Connect.. You *cannot* set this header information in <invoke> commands within a mashups script.

- Within the <variable> statement in a mashup script that also declares the variable.

  If this session attribute does not already exist, the <variable> statement creates a new session attribute. If you assign a default value or construct the variable using literal XML, that value or node set is assigned to the MashZone NextGen session attribute.

You can also use the <assign> command in a mashup script to update or assign values to MashZone NextGen session attributes. The output variable in <assign> simply uses the `$session.`*attribute-name* syntax.

Once MashZone NextGen session attributes exist, you can use them in mashup scripts just like any other variable, using the syntax `$session.`*attribute-name* .

MashZone NextGen session attributes can also hold sets of nodes, unlike user or global parameters. You can use XPath expressions after the attribute name to point to specific nodes within the session attribute.

Using MashZone NextGen Attributes in Mashup Statements

You can refer to MashZone NextGen attributes in any statement where you can refer to variables. For example:

```
...
<variables>
  <variable name="global.defaultContext" type="string"/>
  <variable name="user.firstName" type="string"/>
  <variable name="characters" type="document">
    <customer>
      <id>dog1</id>
      <name>Goofy</name>
      <city>Toon Town</city>
    </customer>
    <customer>
      <id>mouse2</id>
      <name>Minnie</name>
      <city>Toon Town</city>
    </customer>
  </variable>
  <variable name="session.key1" type="string"/>
  <variable name="session.customerInfo" type="document"/>
  <variable name="session.dogInfo" type="document" />
</variables>
...
<invoke service="myService" operation="someQuery"
  inputvariables="$global.defaultContext" .../>
...
<filter filterexpr="/customer[matches(firstname, $user.firstName )] inputvariable="$allCustomers"
...
<assign literal="New York Giants" outputvariable="$session.key1"/>
<assign fromvariable="$characters"
   outputvariable="$session.customerInfo"/>
<assign fromexpr="$session.customerInfo/customer[@id='dog1']"
   outputvariable="$session.dogInfo"/>
```

The <invoke> statement runs a mashable or mashup named myService which will be passed the value of the global attribute name `defaultContext`.

Similarly, the <filter> statement will filter out the $allCustomers variable for those customers whose first name matches the name of the current user.

The last three statements assign values to session attributes for the current user session.

Removing Duplicates With Filtering

Merging, joining or grouping results from several mashables can result in duplicate items in the combined result. You can remove duplicates with the <filter> statement and a filter expression that uses the *axis* feature in XPath to compare preceding or following items. See "XPath Axes" on page 618 for basic information on this XPath feature.

To remove duplicates in a mashup, simply merge, join or group results. If needed, sort the combined results based on the key field that determines uniqueness. This ensures that duplicates are contiguous.

Then use <filter> with a filtering expression that compares the key value of either the preceding or following 'item' to determine if this 'item' is unique. See "Example 91. Unique Filter Example" on page 617 for an example of removing duplicates.

Unique Filter Example

This example merges the results from two RSS mashables and then checks the title of each item to remove duplicates:

```
<mashup xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
    ../xsd/EMMLPrestoSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    name = "MergeFeeds">
  <output name="result" type="document"/>
  <!-- invoke two RSS feeds -->
  <directinvoke outputvariable="$feed1"
    endpoint="http://www.nytimes.com/services/xml/rss/nyt/HomePage.xml" />
  <directinvoke outputvariable="$feed2"
    endpoint="http://www.nytimes.com/services/xml/rss/nyt/World.xml" />
  <!-- merge the results -->
  <merge inputvariables="$feed1, $feed2" outputvariable="$result"/>
  <!-- filter for unique items based on title -->
  <filter inputvariable="$result" outputvariable="$result"
    filterexpr="/rss/channel/item[not(preceding::title = ./title)]" />
</mashup>
```

The filtering expression uses:

- The *not()* XPath function to negate the comparison. It only selects items that do not have any preceding items with matching titles.
- The *preceding* axis to check all previous titles in the merged feeds against the title for the current item.

Because of the structure of RSS results, you could also use `preceding-sibling::item/title`. If you sort the results based on `item/title` you could also simply check just the closest item title with `preceding-sibling::item[1]/title` to rule out duplicates.

- The . in `./title` is the short syntax to identify the current context node. This selects the child `title` for the current context to compare it to all previous titles.

XPath Axes

Axes in XPath are a syntax that allow XPath expressions to refer to other nodes based on a relationship with the node that is the *current context*. Take a simple <filter> statement, such as this:

```
<filter inputvariable="$a" outputvariable="$a" filterexpr="/rss/channel/
item[contains(title,'Java')" />]
```

As the filter is processed, it checks each `item` node and that node becomes the current context. The `title` node in this example, in fact, uses the default XPath axis -- the child axis. Because there is no other axis identifier, the filter looks for `title` as a child of `item`.

There are many other axes in XPath that allow you to refer to previous nodes, following nodes, the parent node, ancestor nodes, descendant nodes and many more. See the XPath 2.0 specification for a complete list of valid axes.

To use a different axis than the default child axis, you add a prefix to each node name in the form:

axis-name `::`*node-name*

For example, `preceding::item` identifies any `item` node that comes before the current node and is not an ancestor. The path expression `ancestor::channel` identifies the `channel` node that is a parent or earlier ancestor at any level of the document to the current node. You can also use wildcards, such as `following::*` or `following::node()` to identify all following nodes of any name.

To filter out duplicates, you typically use one of these axes:

- *preceding*: any nodes from the document root node to the current context node that come before the current context in document order and are *not* ancestor nodes of the current context.
- *following*: any nodes that come after the current context in document order and are *not* descendants of the current context.
- *preceding-sibling*: any nodes that have the same parent as the current context and occur before the current context in document order.
- *following-sibling*: any nodes that have the same parent as the current context and occur after the current context in document order.

Setting the Mashup Response Character Encoding

There are two aspects involved in setting the character encoding for the mashup response:

- Setting the HTTP Content Type Header in the Response
- Setting the Character Encoding in the Response XML Declaration

Setting the HTTP Content Type Header in the Response

To add the HTTP header for content type to the response, simply declare a variable in the mashup script with the appropriate name and value. For example:

```
<variables>
  <variable name="httpResponse.Content-Type" type="string"
    default="text/html; charset=ISO-8859-1"/>
</variables>
```

See "Adding HTTP Headers to the Mashup Result" on page 607 for the full details.

Setting the Character Encoding in the Response XML Declaration

If the response is generally going to be returned in an XML format, the XML declaration `<?xml version="1.0"?>` in the response also has an optional `encoding` property. You can define what character encoding the MashZone NextGen Server uses for the result using the `output-encoding` attribute in the <output> statement. This also sets the `encoding` property in the XML declaration.

```
<mashup name="myMashup"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
     ../xsd/EMMLPrestoSpec.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <output name="result" type="document" output-encoding="ISO-8859-1"/>
...
  <assign fromexpr="$someVar/books/genre[1]"
    outputvariable="$result"/>
</mashup>
```

See "Commonly Supported Output Character Encodings" on page 461 for some of the most commonly used characters encoding values.

Wrapping Results or Variables in CDATA Sections

In some cases, XML content uses CDATA sections to treat element content that may contain markup as though it is text. This is most commonly used to avoid having to escape HTML markup that is inside XML elements to keep the markup intact for later processing. However, it can be applied to any element content that may have HTML or XML markup which you need to escape as an entire block.

Common examples are web feeds using RSS or Atom formats where article titles or descriptions contain HTML. For example:

```
<rss>
  <channel>
  ...
  <item>
    <title><![CDATA[The <b>article</b> title with <span
>markup</span>]]></title>
    ...
  </item>
  ...
</rss>
```

You can wrap the contents of specific elements in CDATA sections in the result of a mashup or in variables. Use the `output-cdata-section-elements` attribute on either the <output> or <variable> declarations.

The value of `output-cdata-section-elements` is the name of one or more elements whose content should be wrapped in a CDATA section. For example:

```
<variable name="postBody" type="document"
    output-cdata-section-elements = "title description" />
```

If the elements in question have namespaces, you must add the namespace in the form {*namespace-URI* } before the element name. For example:

```
<output name="result" type="document"
    output-cdata-section-elements =
      "{http://www.opengis.net/kml/2.2}text
       {http://www.opengis.net/kml/2.2}bgColor" />
```

For common examples of wrapping results in CDATA sections, see:

- "Example 92. CDATA Sections in <output>" on page 620
- "Example 93. CDATA Sections in <variable> Used as Input to Mashables/Mashups" on page 621
- "Example 94. CDATA Sections in <variable> When Casting Strings to Documents" on page 621

CDATA Sections in <output>

With an input document in the following form, the CDATA section in the <text> element is stripped by the parser, but the HTML markup remains:

```
<kml xmlns="http://www.opengis.net/kml/2.2">
  <document>
    <Style id="medSporeCountBalloon">
      <BalloonStyle>
        <bgColor>#EBFFEB</bgColor>
        <text><![CDATA[<h1>Some HTML content</h1>]]></text>
      </BalloonStyle>
    </Style>
  </document>
</kml>
```

With an <output> declaration for the mashup in the form:

```
<output name="result" type="document"
    output-cdata-section-elements =
      "{http://www.opengis.net/kml/2.2}text
       {http://www.opengis.net/kml/2.2}bgColor" />
```

The final result of the mashup wraps the content of both the <text> and the <bgColor> elements in CDATA markers:

```
<kml xmlns="http://www.opengis.net/kml/2.2">
  <document>
    <Style id="medSporeCountBalloon">
      <BalloonStyle>
        <bgColor><![CDATA[#EBFFEB]]></bgColor>
        <text><![CDATA[<h1>Some HTML content</h1>]]></text>
      </BalloonStyle>
    </Style>
  </document>
</kml>
```

CDATA Sections in <variable> Used as Input to Mashables/Mashups

Another common scenario occurs when a mashable or other information source expects an input variable with XML content that contains CDATA sections.

```
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    name="queryPost">
  <output name="result" type="document"/>
  <!-- comma separated list of terms to search for -->
  <input name="terms" type="string"/>
  <variable name="words" type="document" />
  <assign fromexpr="tokenize($terms,',')" outputvariable="words" />
  <constructor outputvariable="postBody"><query/></constructor>
  <foreach items="$words" variable="word">
    <variable name="searchList" type="document"
      cdata-output-section-elements="term"/>
    <appendresult outputvariable="$searchList">
      <term>{$word}</term>
    </appendresult>
    <appendresult outputvariable="$postBody">
      {$searchList}
    </appendresult>
  </foreach>
  ...
  <directinvoke endpoint="$queryService" method="POST" header="$contentType"
    requestBody="$postBody"/>
</mashup>
```

CDATA Sections in <variable> When Casting Strings to Documents

You may also need to wrap element content when the input is a string of XML content that you need to cast to an XML document.

```
<input name="rssString" type="string"/>
<variable name="rssDoc" type="document"
  output-cdata-section-elements = "title description" />
<assign fromvar="$rssString" outputvariable="$rssDoc"/>
```

Web Clipping with a Mashup Script

Web clipping, also sometimes called screen scraping, allows you to treat the HTML from any URL as the result of a mashable that you can filter, combine or otherwise transform in a mashup. A web clipping mashup uses the <directinvoke> statement to retrieve

HTML which the MashZone NextGen Server converts to XHTML, in the `http://www.w3.org/1999/xhtml` namespace.

Note: MashZone NextGen uses TagSoup to convert HTML to XHTML and ensure the response is well-formed. Because of this conversion, the result may not match the HTML source exactly.

In versions 3.7 and earlier, MashZone NextGen used JTidy for this conversion. If needed, you can choose JTidy for backwards compatibility. See "Handling HTML Responses" on page 623 for an example.

You can then use this XHTML in the mashup script as a mashable response.

Example

This example uses the results of a Google query as a web clipping result to output specific links:

```
<mashup name="Ruby"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:res="http://www.myCompany.com/googleQuery"
    xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <operation name="queryGoogle">
    <output name="result" type="document">
      <res:queries xmlns:res="http://www.myCompany.com/googleQuery"/>
    </output>
    <variable name="uri" default="http://www.google.com/search?q=ruby"/>
    <directinvoke outputvariable = "$searchresult" endpoint="$uri"/>
    <foreach variable="query" items="$searchresult//xhtml:a[starts-with(@href, '/url?q=')]">
      <appendresult outputvariable="$result">
        <res:itemlink>
        {attribute href {resolve-uri($query/@href, $uri)}}
        </res:itemlink>
      </appendresult>
    </foreach>
  </operation>
</mashup>
```

The XML result from this mashup looks something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<queries xmlns="http://www.myCompany.com/googleQuery">
 <itemlink href="http://www.ruby-lang.org/"/>
 <itemlink href="http://en.wikipedia.org/wiki/Ruby_programming_language"/>
 <itemlink href="http://en.wikipedia.org/wiki/Ruby"/>
 <itemlink href="http://www.rubyonrails.org/"/>
 <itemlink href="http://www.rubycentral.com/"/>
 <itemlink href="http://www.rubycentral.com/book/"/>
 <itemlink href="http://www.w3.org/TR/ruby/"/>
 <itemlink href="http://www.youtube.com/watch?v=JMDcOViViNY"/>
 <itemlink href="http://www.zenspider.com/Languages/Ruby/QuickRef.html"/>
 <itemlink href="http://poignantguide.net/"/>
...
</queries>
```

Handling HTML Responses

Document type variables in EMML typically contain well-formed XML. Responses from mashables or direct web clipping can return HTML.

With HTML responses, the MashZone NextGen Server converts the response to well-formed XHTML to ensure access to the data. Two issues can arise during parsing:

- Parsing errors for some responses. You can use the `subtype` attribute to handle parsing errors. See "Example 96. Subtype for Parsing Errors" on page 623 for an example.
- Differences in the XHTML result for mashups from 3.7 or earlier.

  The parser that MashZone NextGen uses for conversion to XHTML changed in version 3.8. This can cause errors in mashups created in version 3.7 or earlier.

  You can control which parser is used in the mashup using the `htmlparser` attribute. See "Example 97. JTidy for Backwards Compatible Parsing" on page 623 for an example.

Subtype for Parsing Errors

To prevent parsing errors for HTML responses, use `subtype="HTML"` on <output> or <variable>. For example:

```
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
     xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
      ../schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    name="GoogleWebClipping">
  <input name="query" type="string" default="ruby"></input>
  <output name="result" type="document"/>
  <variable name="searchresult1" type="document" subtype="HTML"/>
  <directinvoke outputvariable = "searchresult1"
      endpoint="http://www.google.com/search?q={$query}"/>
  <foreach variable="query" items="$searchresult1//xhtml:a[@class]">
    <appendresult outputvariable="result">
      <itemlink>{$query/@href}</itemlink>
    </appendresult>
  </foreach>
</mashup>
```

JTidy for Backwards Compatible Parsing

In versions 3.7 and earlier, MashZone NextGen used JTidy to parse HTML responses and convert them to well-formed XHTML. For versions 3.8 and later, MashZone NextGen uses TagSoup for parsing and converting HTML responses.

This change in parser may cause parsing errors or other problems in mashups from version 3.7 and earlier. You can choose to use JTidy to ensure backwards compatibility with `htmlparser = "jtidy"`. For example:

```
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
```

```
     xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
      ../schemas/EMMLSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    name="JTidyParsing">
  <input name="query" type="string" default="scala"/>
  <output name="result" type="document"/>
  <directinvoke outputvariable = "searchresult" htmlparser="jtidy"
      endpoint="http://www.google.com/search?q={$query}"/>
  <foreach variable="query" items="$searchresult//xhtml:h3[@class='r']/xhtml:a">
    <appendresult outputvariable="result">
      <itemlink>{$query/@href}</itemlink>
    </appendresult>
  </foreach>
</mashup>
```

Handling JSON Responses or Inputs

Some web services return complex results in the JavaScript Object Notation (JSON) format. Some may also expect complex input parameters in the JSON format. There are two possible solutions to work with JSON data in mashups or macros:

- Pass or receive JSON data as strings and convert the JSON string to an XML document. Once converted to XML, you can use any EMML statements to work with the data.

  The conversion from JSON to XML can *fail*, however, due to discrepancies between the valid syntax of each. Some common problem areas include valid names in JSON that are not valid names in XML and arrays, which JSON supports but XML does not.

  See "Convert JSON to XML" on page 625 for an example of this solution.

- Work with JSON data as an object with limitations to what EMML statements work with the data.

  Input parameters, variables or the output parameter for a mashup or macro can have a datatype of `document` and a subtype of `JSON`, such as:

  ```
  <input name="jsonInput" type="document" subtype="json"/>
  <variable name="jsonVar" type="document" subtype="json"/>
  <output name="jsonResult" type="document" subtype="json"/>
  ```

  Parameters or variables with a `JSON` subtype, can be used directly *only* in these two EMML statements:

  - `<directinvoke>`: as the request body to send to a web service, when the method is POST, or as the outputvariable to receive the response of the invocation. See "Post JSON to a Web Service Using <directinvoke>" on page 625 for an example.

  - `<script>`: as an input variable or the output of the script, with JavaScript as the scripting language.

    With <script>, you can directly manipulate the JSON object using JavaScript methods. See "Accept JSON as Input and Modify with <script>" on page 626 for an example.

Convert JSON to XML

To convert JSON data to XML, a mashup or macro receives the JSON object as a string, converts the JSON string to an XML string and finally converts the XML string to an XML document.

In this example, the result of a mashable is a JSON object which is placed in a string variable, `jsonString`. This is converted to an XML string with <script> and the `toXML` method in the `com.jackbe.presto.common.DataTransformer` utility class in MashZone NextGen:

```
...
  <variable name="jsonString" type="string"/>
  <variable name="xmlStr" type="string"/>
  <variable name="xmlDoc" type="document"/>
  <!-- invoke web service that returns JSON -->
  <invoke service="jsonOutput" operation="Invoke"
    outputvariable="$jsonString" />
  <!-- convert JSON string to XML string -->
  <script type="text/javascript" inputvariables="jsonString"
    outputvariable="xmlStr">
    <![CDATA[
      xmlStr = toJson(jsonString);
      function toJson(str){
          return Packages.com.jackbe.presto.common.DataTransformer.jsonToXml(str);
      }
    ]]>
  </script>
  <!-- cast XML string to document -->
  <assign fromvariable="$xmlStr" outputvariable="$xmlDoc" />
  <!-- proceed with more EMML processing -->
...
```

Once the data is in XML as a string, it is converted to an XML document using <assign> and a document-type output variable. At this point, the data is in a form that the mashup or macro can use any EMML statement needed for any further processing.

Post JSON to a Web Service Using <directinvoke>

For web services that require JSON input parameters, you can use <directinvoke> and the POST method with a variable that has `subtype="JSON"`, such as this example:

```
...
  <variable name="postBody" type="document" subtype="JSON">
    <![CDATA[
      {"locations": [
        { "city": "San Francisco", "lat": "37.7668", "long": "-122.3959" },
        { "city": "Albuquerque", "lat": "35.151317", "long": "-106.539867"},
        { "city": "Seattle", "lat": "47.61132", "long": "-122.320637"}
      ]}
    ]]>
  </variable>
  <directinvoke endpoint="http://someOrg.com/jsonService" method="POST"
    requestbody="$postBody" outputvariable="svcResponse"/>
...
```

In this example, the JSON is defined directly in the variable, but you can also use <assign> or <constructor> to assemble or build a JSON object.

You can also set the datatype for the response from the web service to JSON, simply by declaring the datatype and subtype of the variable to receive the response:

```
...
  <variable name="postBody" type="document" subtype="JSON">
    <![CDATA[
      {"locations": [
        { "city": "San Francisco", "lat": "37.7668", "long": "-122.3959" },
        { "city": "Albuquerque", "lat": "35.151317", "long": "-106.539867"},
        { "city": "Seattle", "lat": "47.61132", "long": "-122.320637"}
      ]}
    ]]>
  </variable>
  <variable name="svcResponse" type="document" subtype="JSON"/>
  <directinvoke endpoint="http://someOrg.com/jsonService" method="POST"
    requestbody="$postBody" outputvariable="svcResponse"/>
...
```

Accept JSON as Input and Modify with <script>

If you need to manipulate a JSON variable in a mashup or macro without converting the data to XML, you must use JavaScript in the <script> statement. You can also accept a JSON object as an input parameter, such as this example macro:

```
...
  <macro name="getFirstJsonCity" >
    <output name="cityStats" type="document" subtype="JSON"/>
    <input name="jsonCities" type="document" subtype="JSON"/>
    <script type="text/javascript" inputvariables="jsonCities"
      outputvariable="cityStats">
      <![CDATA[
        var jsonObj = JSON.parse(jsonCities);
        cityStats = JSON.stringify(jsonObj.locations[1]);
      ]]>
    </script>
  </macro>
...
```

This example is a macro, but you can use these same techniques in a mashup also.

Wrapping POJO Classes with Mashups

You can use any Java class accessible to the MashZone NextGen Server within the mashup <script> statement using JavaScript. Because of this feature, you can easily use POJO classes as MashZone NextGen mashables by wrapping them in mashup scripts.

Note: Based on the JDK version used with the application server that hosts the MashZone NextGen Server, Java language classes are also available within mashup <script> statements.

To wrap POJO classes in mashup scripts

1. Include the POJO classes in the MashZone NextGen Server classpath by copying the compiled class file or JAR file to one of these folders:
   - The external configuration folder, if any, for the MashZone NextGen Server.

Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

- *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
- *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

2. Create a separate wrapper mashup script for each POJO method that you want to expose as a MashZone NextGen mashable information source.
3. In the wrapper mashup script:
   - Add <input> declarations, if needed, to accept input parameters for the Java method and an <output> declaration to receive the method return, if any.
   - Add a <script> statement to invoke the Java code. See "Using Java Classes in a <script> Statement" on page 627 for instructions and examples.

Using Java Classes in a <script> Statement

For JavaScript scripts, you must use the fully qualified class name starting with the keyword `Packages.` when you refer to a Java class. You create an instance of a Java object with `new`, just like any other JavaScript object. For example:

```
<script type="text/javascript">
<![CDATA[
  var list = new Packages.java.util.ArrayList();
  list.add("one");
  list.add("six");
  list.add("three");
  var listLength = list.size();
  print(listLength);
]]>
</script>
```

You can assign properties and call methods on Java instances. See the Rhino Scripting topics for additional techniques and information.

You can also convert Java instances to an XML DOM tree and assign it to a mashup variable. For example:

```
<variable name="xmlDoc" type="document"/>
<script type="text/javascript" outputvariable="$xmlString">
  <![CDATA[
    var dt = new Packages.com.myCompany.test.Address();
    // populating dummy values;
    dt.street = 'disney avenue';
    dt.city = 'toon town';
    dt.zip.code = '91010';
    // convert JavaObject to XML DOM tree;
    xmlString=Packages.com.jackbe.jbp.transform.DataTransformer.javaToXml(dt);
  ]]>
</script>
<!-- use assign to convert from DOM string to document -->
<assign fromvariable="$xmlString" outputvariable="$xmlDoc"/>
```

Example

This example uses a JavaScript script to wrap a Java method with the following signature in a mashup:

```
public String getCategory(String item)
```

```
<mashup name="GetItemCategory"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML
       ../xsd/EMMLPrstoSpec.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<input name="itemId" type="string"/>
<output name="result" type="string"/>
<script type="text/javascript" inputvariables="$itemId"
   outputvariable="$result">
  <![CDATA[
    var item = new Packages.com.myCompany.somePackage.Categories();
    var result = item.getCategory(itemId);
  ]]>
</script>
</mashup>
```

Wrapping WSDL Web Services That Return HTML Results

MashZone NextGen currently does not support HTML within mashable responses unless it is enclosed in a CDATA section or explicitly defined as XML content for the response. Some RSS web feeds, for example, embed HTML inside the <description> element for each item, such as:

```
<description>
  <![CDATA[<p>some HTML here to display</p>]]>
</description>
```

For SOAP web services that return HTML without a CDATA section, you can wrap them in mashup scripts to successfully handle the response. This wrapper mashup uses the sample mashup `SOAPService` in MashZone NextGen to invoke the actual SOAP web service.

To wrap a web service in a mashup script

1. If you have not already published the MashZone NextGen sample mashups, publish the sample `SOAPService` mashup script to make this service available. You can open the `SOAPService` mashup script in Mashup Editor and save it.

   The `SOAPService` mashup is a generic mashup script that can invoke a SOAP service that you specify as input to the mashup. You must also construct the body of the message and any headers needed.

2. Create a new mashup script as the wrapper for this web service and add <output> to the script.

   ```
   <mashup name="WrapMyWSDL"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
         ../xsd/schemas/EMMLPrestoSpec.xsd"
       xmlns="http://www.openmashup.org/schemas/v1.0/EMML">
   ```

```
<output name="result" type="document"/>
</mashup>
```

3. Define variables for these input parameters:

   - *soapEndpoint*: the URL to the endpoint for this SOAP web service.
   - *soapAction*: the URL or other name for the operation you want to invoke in this SOAP web service.
   - *soapHeader*: if there are no SOAP headers needed, you define an empty variable. If you need to send SOAP headers in the request, use <constructor> to create the appropriate headers instead.

   The mashup script should now look something like this:

```
...
<output name="result" type="document"/>
<variables>
  <variable name="soapEndpoint" type="string"
      default="http://www.serviceProvider.com/someService/"/>
  <variable name="soapAction" type="string"
      default="http://www.serviceProvider.com/someService/ActorQuery"/>
  <variable name="soapHeader" type="document"/>
</variables>
</mashup>
```

4. Construct the body of the SOAP request using <constructor>.

   Add the literal XML for the body of the request and fill in values with literals or use dynamic mashup expressions.

   The <constructor> statement would look something like this:

```
...
<constructor outputvariable="$soapPayload">
<someService xmlns="http://serviceProvider.com/WebServices">
  <ActorQuery>Meg Ryan</ActorQuery>
</someService>
</constructor>
</mashup>
```

5. Invoke `SOAPService` using the variables to complete these input parameters *in order*:

   - *soapEndpoint*: the URL of the SOAP web service to invoke.
   - *soapHeader*: any SOAP headers to include in the request.
   - *soapBody*: the payload of the request.
   - *soapAction*: the operation to invoke in this SOAP web service.

   The <invoke> statement should look something like this:

```
...
<invoke service="SOAPService" operation="Invoke" inputvariables="soapEndpoint, soapHeader, soa
</mashup>
```

6. If needed, extract the HTML from the results using <assign>.

   If you want only the HTML from the web service response, you must extract those nodes from the service result. The HTML from the response *must* be well-formed

and fully enclosed in one root node if you are going to use it as the output from the mashup script.

Note: The HTML in the service response is treated as XML. Any XML delimiter characters that are in the HTML content, such as `&`, will use XML escape entities, such as `&`.

The complete wrapper mashup script would look something like this:

```
<mashup name="WrapMyWSDL"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
      ../xsd/EMMLPrstoSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML">
<output name="result" type="document"/>
<variables>
  <variable name="soapEndpoint" type="string"
      default="http://www.serviceProvider.com/someService/"/>
  <variable name="soapAction" type="string"
      default="http://www.serviceProvider.com/someService/ActorQuery"/>
  <variable name="soapHeader" type="document"/>
</variables>
<constructor outputvariable="$soapPayload">
<someService xmlns="http://serviceProvider.com/WebServices">
  <ActorQuery>Meg Ryan</ActorQuery>
</someService>
</constructor>
<invoke service="SOAPService" operation="Invoke" inputvariables="soapEndpoint, soapHeader, soa
<assign outputvariable="$result" fromexpr="$result//*:ActorQueryResponse"/>
</mashup>
```

Pre-Processing/Post-Processing Mashables with Mashups

Pre-processing and post-processing for mashables commonly involves tasks such as decoding encrypted data, encrypting data, decompressing or compressing data, filtering data or data scrubbing. You can handle these types of requirements with a mashup that uses scripting to invoke pre- or post-processing code.

Tip: It is a good practice to design pre- or post-processing code as its own service that can be invoked by many mashups.

Pre-processing Mashup

To implement pre-processing

1. Write the logic to handle pre-processing as a Java class or use Java libraries, as needed.
2. Add the custom Java classes to the classpath for the MashZone NextGen Server. Copy them to:
   - The external configuration folder, if any, for the MashZone NextGen Server.

Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

- *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
- *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

3. Restart the MashZone NextGen Server.
4. Create a mashup script with the following components:
   a. <input> variables, as needed, to receive the data to be pre-processed and an <output> variable for the result. For example:

```
...
<operation name="getreviews">
  <input name="userId" type="string" default="someone@myOrg.com" />
  <input name="accessId" type="string"
    default="xOtPy9Sy7JCI3Y8aNqEkZBxmkPQw/ZH8"/>
  <output name="result" type="string"/>
```

   b. Add a <script> statement to call the pre-processing logic.

      This example uses JavaScript scripting and shows an encryption class that decrypts the accessId input parameter. To call the pre-processing logic in JavaScript, you must use fully qualified class names starting with the `Packages` keyword.

```
...
  <output name="result" type="string" inputvariables="$accessId"
    outputvariable="$accessId"/>
  <script type="text/javascript">
    <![CDATA[
      var encrypter = new
        Packages.com.myOrg.services.Encrypter("My Pass Phrase!")
      accessId = encrypter.decrypt(accessId);
    ]]>
  </script>
```

   c. Invoke the mashable information source with the pre-processed data. For example:

```
...
  </script>
  <invoke service="AmazonREST" operation="getData"
    inputvariables="'AWSECommerceService', '2007-07-16', accessId,
       'ListSearch', 'WishList', userId"
    filterexpr="//ns:List[ns:TotalItems > 1]"
    outputvariable="$result"/>
```

Post-processing Mashup

At its simplest, any mashup that invokes a mashable and performs an action using the result is providing post-processing.

To implement post-processing logic that is directly provided by mashup script statements

1. Write the logic to handle the post-processing as a Java class or use Java libraries, as needed.
2. Add the custom Java classes to the classpath for the MashZone NextGen Server. Copy them to:
   - The external configuration folder, if any, for the MashZone NextGen Server.

     Important: Deploying additional resources, such as custom XPath classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

   - *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
   - *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
3. Restart the MashZone NextGen Server.
4. Create a mashup script with the following components:
   a. <variable>s , as needed, to receive the results from the mashable or other data to be post-processed and an <output> variable for the result. For example:

```
...
<operation name="encryptResult">
  <variable name="svcResult" type="document"/>
  <output name="result" type="string"/>
```

   b. Invoke the mashable. For example:

```
...
  <output name="result" type="string"/>
  <invoke service="AmazonREST" operation="getData"
    inputvariables="'AWSECommerceService', '2007-07-16',
       'xOtPy9Sy7JCI3Y8aNqEkZBxmkPQw/ZH8', 'ListSearch', 'WishList',
       'someone@myOrg.com'"
    filterexpr="//ns:List[ns:TotalItems > 1]"
    outputvariable="$svcResult"/>
  <constructor outputvariable="$svcResult">
    <ns:mylist>{$svcResult/ns:ListSearchResponse/ns:Lists/ns:List}</ns:mylist>
  </constructor>
```

5. Add a <script> statement to call the post-processing logic.

   This example uses JavaScript scripting. It shows an encryption class that encrypts the filtered and constructed results from the mashable. To call the post-processing logic in JavaScript, you must use fully qualified class names starting with the `Packages` keyword.

```
...
  </constructor>
  <script type="text/javascript" inputvariables="$svcResult"
    outputvariable="$result">
    <![CDATA[
      var encrypter = new
```

```
        Packages.com.myOrg.services.Encrypter("My Pass Phrase!")
      result = encrypter.encrypt(svcResult);
    ]]>
  </script>
...
```

Self-Joins with a Single Dataset

Self-joins compare values in repeating nodes from a single result to create the join result. They typically resolve cross references within a single result (one item refers to another item in the same dataset) using the <join> statement.

Note: The example shown in this topic resolves references to the supervisor of employees (the repeating items) that have a supervisor where the supervisors are themselves employees (and thus in that same set of repeating items).

To do this, the join condition for self-joins compare values between repeating items in the same variable. However, EMML join conditions do not support comparisons within a single variable, such as the following join condition:

```
...
  <join outputvariable="result"
joincondition="$staffDoc/employees/employee/supervisor =
$staffDoc/employees/employee/id" >
    <select name="employees">
      <result>
        <employee>{$staffDoc/name/string()}</employee>
        <supervisor>{$staffDoc/name/string()}</supervisor>
      </result>
    </select>
  </join>
..
```

This example selects employees who have a supervisor and joins employee data for that supervisor to the employee's data using the single `$staffDoc` variable.

The solution to get a self-join is to copy the single results you want to use in the join condition to a second variable. Then let the join condition compare node values from two different variables.

For example:

```
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
      ../schemas/EMMLSpec.xsd"
      xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
      name="SelfJoin">
  <output name="result" type="document"/>
  <input name="staffDoc" type="document">
    <employees>
      <employee>
        <id>100</id>
        <name>Dmitry</name>
        <supervisor>101</supervisor>
      </employee>
      <employee>
        <id>101</id>
        <name>Patricia</name>
      </employee>
```

```
      <employee>
        <id>102</id>
        <name>Blaine</name>
        <supervisor>101</supervisor>
      </employee>
      <employee>
        <id>103</id>
        <name>Randy</name>
        <supervisor>101</supervisor>
      </employee>
      <employee>
        <id>104</id>
        <name>Harold</name>
        <supervisor>101</supervisor>
      </employee>
      <employee>
        <id>105</id>
        <name>Merle</name>
        <supervisor>102</supervisor>
      </employee>
    </employees>
  </input>
  <!-- duplicate result set in second variable -->
  <variable name="staffMirrorDoc" type="document" />
  <assign fromvariable="$staffDoc" outputvariable="$staffMirrorDoc" />
  <!-- join employee supervisor data to employees -->
  <join outputvariable="result"
      joincondition="$staffDoc/employees/employee/supervisor =
        $staffMirrorDoc/employees/employee/id" >
    <select name="employees">
      <result>
        <employee>{$staffDoc/name/string()}</employee>
        <supervisor>{$staffMirrorDoc/name/string()}</supervisor>
      </result>
    </select>
  </join>
</mashup>
```

Using XQuery for Outer Joins

The <join> command in EMML works like an inner join for SQL - only items that have values that match the join condition are selected. For outer joins, you want to see all items from one set of results and join this with other results even if there is no matching result.

You can get the affect of an outer join using XQuery and the <constructor> statement to build the join results.

Sample Mashable Results

This sample of an outer join uses three "results" from mashables that are shown for clarity as variables:

Suppliers Results

```
<variable name="suppliers" type="document">
  <suppliers>
    <supplier>
      <supplierId>s1</supplierId>
```

```
      <supplierName>Genoa and Sons</supplierName>
    </supplier>
    <supplier>
      <supplierId>s2</supplierId>
      <supplierName>Angora, Ltd.</supplierName>
    </supplier>
    <supplier>
      <supplierId>s3</supplierId>
      <supplierName>Desideratti</supplierName>
    </supplier>
  </suppliers>
</variable>
```

Items Results

```
<variable name="items" type="document">
  <items>
    <item>
      <partId>p1</partId>
      <supplierId>s1</supplierId>
      <price>10</price>
    </item>
    <item>
      <partId>p2</partId>
      <supplierId>s1</supplierId>
      <price>10</price>
    </item>
    <item>
      <partId>p3</partId>
      <supplierId>s1</supplierId>
      <price>10</price>
    </item>
    <item>
      <partId>p2</partId>
      <supplierId>s2</supplierId>
      <price>11</price>
    </item>
  </items>
</variable>
```

Parts Results

```
<variable name="parts" type="document">
  <parts>
    <part>
      <partId>p1</partId>
      <description>Dupioni Silk, grey/blue</description>
    </part>
    <part>
      <partId>p2</partId>
      <description>Angora wool, cream</description>
    </part>
    <part>
      <partId>p3</partId>
      <description>Cashmere, pink</description>
    </part>
    <part>
      <partId>p4</partId>
      <description>Paisley, gold and blue</description>
    </part>
    <part>
      <partId>p5</partId>
      <description>Georgette, white</description>
```

```
    </part>
  </parts>
</variable>
```

Constructing the Join

Rather than using <join>, you use <constructor> to construct the result and XQuery to select from the results to create the join:

The <constructor> creates a root node, `<result>`, as literal XML and uses a dynamic mashup expression with XQuery inside to select all suppliers in the `$suppliers` variable and sort them by name:

```
<constructor outputvariable="$result">
  <result>
  {
    for $s in $suppliers/suppliers/supplier
    order by $s/supplierName
...
```

The XQuery expression returns a `<supplier>` node for each supplier selected:

```
...
  <result>
  {
    for $s in $suppliers/suppliers/supplier
    order by $s/supplierName
    return
    <supplier>
...
    </supplier>
  }
  </result>
</constructor>
```

Within this literal `<supplier>` node, another dynamic mashup expression with an XQuery expression selects the matching items and parts, if any, to create the join:

```
...
    <supplier>
      {
        $s/supplierName,
        for $i in $catalog/items/item[supplierId = $s/supplierId],
            $p in $parts/parts/part[partId = $i/partId]
        order by $p/description
        return $p/description
      }
    </supplier>
...
```

In the final result, each `<supplier>` node has a `<supplierName>` node. If there are matching items and parts, this is followed by `<description>` nodes for the parts, sorted in description order. If no items or parts match a supplier, there are no `<description>` node children.

The complete <constructor> statements looks like this:

```
<constructor outputvariable="$result">
  <result>
  {
    for $s in $suppliers/suppliers/supplier
    order by $s/supplierName
```

```
    return
    <supplier>
      {
        $s/supplierName,
        for $i in $catalog/items/item[supplierId = $s/supplierId],
            $p in $parts/parts/part[partId = $i/partId]
        order by $p/description
        return $p/description
      }
    </supplier>
  }
  </result>
</constructor>
```

Defining and Using Custom Mashup Statements with Macros

You can create user-defined statements for use in mashup scripts or macros using the <macro> statement in EMML. Macros can also be used as custom action blocks to extend mashup capabilities in Wires.

Macros are basically snippets of mashup logic that can accept input parameters and produce output or exceptions. They can do anything that you can do in a mashup, but they can only be used in a mashup or another macro.

Note: In previous releases, macros could not call other macros nor could macros be defined within macros. These limitations no longer apply.

There are two ways to create macros depending on how you need to use them:

- *Inner macros* are defined in a single mashup or macro. You can only use inner macros in the mashup or macro where they are defined.

  See "<macro>" on page 638 for basic instructions on defining a macro.

- *Reusable macros* are defined in macro libraries so that they can be used in any mashup or macro. See "<macros>" on page 645, "Creating and Registering Macros" on page 643 and "Including Macro Domains in Mashup Scripts or Macros" on page 646 for instructions.

Once you have defined macros, you use them in mashup scripts or other macros much like any other mashup statement. See "<macro:custom-macro-name>" on page 650 and "Calling a Macro" on page 647 for instructions.

You can also provide metadata for macros using <presto:macro-meta>. This metadata determines whether a macro can be used as a custom Wires block. If so, it can also provide configuration for the custom block. See "Adding Custom Blocks to MashZone NextGen Wires Using Macros" on page 392 for more information on adding macros to Wires and configuring them as custom action blocks.

The MashZone NextGen Server also has some built-in macros that provide additional capabilities for mashups. See "Generating Mashup Scripts Dynamically" on page 654 for information on these built-in macros.

<macro>

One macro definition in a mashup script or another macro, or by itself or in a macro library. Macros define custom statements that can be used in mashup scripts, another macro or as custom blocks in Wires.

Note: Limitations in previous releases that prevented macros from calling or declaring other macros no longer apply.

Macros can contain any EMML declaration and any EMML statement.

Macros allow behavior to be parameterized. They support both input parameters and an output parameter. Macros have access *only* to the variables defined within the macro plus their own input parameters.

Access to macros depends on where they are defined:

- Macros declared in a mashup script or another macro are *inner macros* that are only accessible within that mashup or macro.
- Macros declared individually or in a macro library are *reusable macros* that can be used in any mashup script or macro that includes the domain for that macro. See "Including Macro Domains in Mashup Scripts or Macros" on page 646 for more information.

  Macros can also be used to define custom blocks for Wires. See "Adding Custom Blocks to MashZone NextGen Wires Using Macros" on page 392 for more information.

  Reusable macros can also be defined as global macros that have no domain, but are accessible in any mashup script or macro. See "Creating and Registering Macros" on page 643 for more information on global macros and macro domains.

Macros must have a name. You then add any EMML statements or declarations needed to accomplish the processing that the macro performs. For more information and examples, see:

- "Example 101. Macro Names" on page 639
- "Example 103. Macro Inputs and Output" on page 640
- "Example 104. Variable Access" on page 641
- "Example 105. Macro Inputs and Dynamic Mashup Expressions" on page 641
- "Example 106. Macro Metadata" on page 641
- "Example 107. Macro Logic" on page 642
- "Calling Macros From Another Macro" on page 649

Can Contain	(Declarations Group | Variables Group | Macroincludes Group | presto:presto-meta | presto:macro-meta | Statements Group | macro:*custom-macro-name* |

	presto:beginTransaction | presto:commitTransaction | presto:rollbackTransaction)+
Allowed In	mashup | operation | macros | macro

Attributes

Name	Required	Description
name	yes	The name for this macro definition. This becomes the custom statement name within the calling EMML script or macro, so macro names must be valid XML names. This is also the default block name if the macro is used as a custom block in Wires. Macro names *must* be unique within the domain they belong to. They can contain ASCII letters, numerals, periods (.), underscores (_) or dashes (-). Names must start with a letter.

<macro> Examples

Macro Names

Add a <macro> element and define the name of the custom statement in the `name` attribute, such as this:

```
<macro name="myCustomMacro"></macro>
```

Macro names define the name of the custom statement that you add to the mashup script, so they must follow XML rules for a valid name:

- They must begin with an ASCII letter
- They can contain ASCII letters and numbers, periods (.), underscores (_) and dashes (-).
- They *cannot* contain spaces or other symbols or punctuation.

Macro names must be unique within the domain they belong to.

Important: Macros are defined as EMML statements in the *macro reference namespace* to ensure that they are separate and unique from EMML declarations and statements that belong to the mashup namespace. See "Calling a Macro" on page 647 for more information.

Namespaces in Macros

You must add any namespaces that are used within a macro to the <macro> element, similar to declaring namespaces on <mashup>. You should add namespaces:

- Used within the data or logic of the macro.
- If the macro calls another macro. Add the Macro Reference namespace.
- If the macro is designed as a custom block for Wires. Add the MashZone NextGen Mashups Extensions namespace.

See "EMML Namespaces" on page 450 for the current Macro Reference namespace and MashZone NextGen Mashups Extensions namespace. For example:

```
<macros xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/ ../xsd/EMMLPrestoSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    domain="myDomain">
...
  <macro name="mySalesForceTest"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
      xmlns:urn="urn:enterprise.soap.sforce.com"">
    ...
    <constructor outputvariable="$contactrequest">
      <urn:query prefix="urn" uri="urn:enterprise.soap.sforce.com">
        <urn:queryString>{$sf_where}</urn:queryString>
      </urn:query>
    </constructor>
    <invoke service="salesforce" operation="query"
        inputvariables="contactrequest"
        outputvariable="$sf_result"
        header="$requestheader"/>
    ...
  </macro>
...
</macros>>
```

Macro Inputs and Output

Typically, you define some input parameters for the macro, although this is not required. If the macro returns results, you must also define an output parameter.

For example:

```
...
<macro name="conditionalInvoke"
      xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions
      xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro" >
  <input name="sid" type="string"/>
  <input name="oid" type="string"/>
  <input name="inputs" type="string"/>
  <input name="condition" type="string"/>
  <output name="macroResult" type="document"/>
</macro>
...
```

See "Declaring Mashup and Macro Variables and Parameters" on page 452 for more information on variables and parameters for macros.

If the macro will be used as a custom block in Wires, you can also use <presto:macro-meta> to control the look and behavior for the property fields in Wires corresponding to macro input parameters. See "Example 106. Macro Metadata" on page 641 for more information and additional links.

Variable Access

Both inner and reusable macros have access *only* to their input parameters and any variables that are declared within the macro.

The output parameter (<output>) declared in the macro is *not* accessible by name outside the scope of the macro. So for the example shown previously, references to `$macroResult` outside of the macro would not return the macro result. However, the results of the macro are automatically assigned to the output variable that is identified when the macro is called. See "<macro:custom-macro-name>" on page 650 for more information.

Macro Inputs and Dynamic Mashup Expressions

Like generic mashup scripts, macros may need to allow calling mashups or macros to send XPath expressions or variables as input parameters that are then used within statements in the macro.

To support passing XPath expressions in a macro input parameter, you must use "Dynamic Mashup Expressions" on page 604 to refer to these parameters within the macro.

Macro Metadata

All reusable macros that have been registered in MashZone NextGen can be used in EMML for mashup scripts or macros created in the Mashup Editor or an XML editor. Reusable macros are also visible as custom action blocks in Wires for use in mashups that users create graphically, *unless* you disable this with the MashZone NextGen extension statement <presto:macro-meta>.

To disable a macro as a custom action block for Wires, set the block usage to `System`, such as this example:

```
<macro name="myCustomMacro">
  <presto:macro-meta>
    <block usage="System"/>
  </presto:macro-meta>
</macro>
```

For macros that are visible as custom blocks in Wires, you can use <presto:macro-meta> to provide additional information, such as help, or to configure the look and behavior of the custom block in Wires. For more information and links on configuring custom blocks, see "Configure Properties for Custom Blocks" on page 403.

Macro Logic

Within the <macro> element, you can use any EMML declaration or statement to define macro behavior including <macro>, to define an *inner* macro, <include> or <macro:*name* domain="*domain-name* ">, to call another macro.

The following example is a macro to invoke any registered mashable in MashZone NextGen if a condition is met:

```
...
<macro name="conditionalInvoke"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"> >
  <input name="sid" type="string"/>
  <input name="oid" type="string"/>
  <input name="inputs" type="string"/>
  <input name="condition" type="string"/>
  <output name="macroResult" type="document"/>
  <if condition="{$condition}">
    <invoke service="$sid" operation="$oid"
        inputvariables="inputs" outputvariable="$macroResult"/>
  </if>
</macro>
...
```

This example uses a dynamic mashup expression within the `condition` attribute for <if> to ensure that the XPath expression defined in the `condition` input parameter is evaluated.

The next example adds latitude and longitude data to an address based on the Yahoo! Maps mashable which can then be used to display mashable results in a map.

```
<macro name="AnnotateMacro"
    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
    xmlns:y="urn:yahoo:maps">
  <!-- location document should be <address> root with <city>, <state> and
       <zip> children -->
  <input name="location" type="document" />
  <output name="macroResult" type="document"/>
  <variables>
    <variable name="locationstr" type="string"/>
  </variables>
  <!-- join address input into a string -->
  <assign fromexpr="string-join(($location//city, $location//state,
    $location//zip), ',')" outputvariable="$locationstr"/>
  <!-- call Yahoo maps -->
  <directinvoke endpoint="http://local.yahooapis.com/MapsService/V1/geocode"
                appid="YahooDemo"
                output="xml"
                location="$locationstr"
                outputvariable="$georesult"/>
  <!-- add geographic result to location data -->
  <annotate variable="$location" expr="." >
    element geo:lat { $georesult//y:Latitude/string() },
    element geo:long { $georesult//y:Longitude/string() }
  </annotate>
  <!-- put annotated location data in macro output -->
  <assign outputvariable="$macroResult" fromvariable="$location"/>
</macro>
```

Creating and Registering Macros

Inner macros can be defined within mashup scripts or other macros and can be used only within the containing mashup or macro. You do not have to register inner macros.

Reusable macros can be used in any mashup script or other macro that is registered in the MashZone NextGen Server where the macro is registered. Reusable macros can also define custom action blocks in Wires for use in mashups that are created graphically.

You register individual reusable macros or entire macro libraries as either *global* or in a specific *domain*. Global macros have no domain. Global macros also include the built-in macros for MashZone NextGen and are always available for use in mashups or other macros.

Domains let you organize macros into useful categories and include related sets of macros easily in mashup scripts or other macros. Domain names also ensure that macro names are unique. You must explicitly include domains in mashup scripts or other macros in order to use macros from that domain. For macros that you use as custom action blocks in Wires, macro domains define the menu category where the custom block appears in Wires.

You can create individual inner or reusable macros, and register reusable macros in Mashup Editor. See "Create an Individual Reusable Macro" on page 643 for instructions.

You can also define one or several reusable macros in a *macro library* and register macros in bulk using mashup utilities. Macro libraries are XML files that define a macro domain and the reusable macros for that domain. See "Create a Reusable Macro Library" on page 645 for instructions.

Create an Individual Reusable Macro

1. To define a macro
   - That is an *inner macro*, start a new mashup or open an existing mashup in the Mashup Editor. Add a <macro> statement from the **Actions** menu on the appropriate line.

     Inner macros defined in a mashup script can only be used within that mashup.
   - That is *reusable* in any mashup or in other macros, expand the **Macros** menu in the Mashup Editor and click **Add New Macro**.
2. Enter a name for the macro in the `name` attribute on <macro>.

   ```
   <macro name="myMacro">
     ...
   </macro>
   ```

   Macro names must be valid XML names as they become the element name for the custom EMML statement defined by the macro. They must also be *unique* within the domain in which you register them.

3. If this macro should be used as a custom block in Wires, add the MashZone NextGen Mashups Extensions namespace to the <macro> element.

   Note: See "EMML Namespaces" on page 450 for the current MashZone NextGen Extensions Namespace.

   For example:

```
<macro name="myMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
  ...
</macro>
```

4. To allow this macro to call other macros, add the macro reference namespace to the <macro> element.

   See "EMML Namespaces" on page 450 for the current Macro Reference Namespace.

   For example:

```
<macro name="myMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
  ...
</macro>
```

5. To allow this macro to call macros from other domains, add <include> elements identifying each domain. See "Including Macro Domains in Mashup Scripts or Macros" on page 646 for instructions.

   Note: This step is not required for global macros.

6. Add <input> and <output> parameters, other EMML statements or calls to other macros to define the logic of the macro. See "<macro>" on page 638 for details and examples.

7. For reusable macros, click **Save As** to save and register the macro.

   a. Enter the name for the macro.

   b. Enter a **Domain Name** to register this macro to a specific domain. If you leave the domain blank, the macro is registerd as a global macro.

      Domain names ensure that macro names are unique. For macros designed to be custom blocks in Wires, the domain also defines the menu category where the custom block appears. Domain names:

      - Must be unique.
      - Are case sensitive.
      - Can contain letters, numbers, periods (.), dashes (-) and underscores (_).

   c. Click **Save**.

The **Macros** menu updates and lists any new macros that are defined within a domain.

Create a Reusable Macro Library

Macro libraries are XML files with the definitions or one ore several macros within one domain. Because they are file-based, macro libraries can be a good way to track macros in source control but still easily register them in MashZone NextGen.

To create a macro library file

1. Create a macro library file in any XML editor of your choice.
2. Add a <macros> element as the root element of the file.

   For example:

```
<macros xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  domain="myDomain">
...
</macros>>
```

3. Optionally, define the domain for all macros in this library in the `domain` attribute.
4. Define one or more macros for this library with the <macro> statement.
5. Save the macro library file.

   Macro libraries typically use the *.emml-macros file extension, although this is not required.
6. Use the `macropub` mashup utility command to register specific macros or all macros in this library. See "Publishing and Managing Macros from the Command Line" on page 668 for instructions.

<macros>

The root element for a file containing one or more *reusable* macro definitions that may be used in many mashup scripts or macros. By default, each MashZone NextGen Server has a *global* domain with built-in macros. You can also publish macros from your own macro libraries to the MashZone NextGen Server and define your own domains.

See "Creating and Registering Macros" on page 643 for more information and examples.

Can Contain	(<macro>)+
Allowed In	Root element with no parents.

Attributes

Name	Required	Description
domain		The name of the purpose or domain for all macros in this macro library. Domains are used

Name	Required	Description
		to categorize macros in MashZone NextGen and define the menu category when macros are used as custom blocks in Wires. They also ensure that macro names are unique across different domains. **Note:** Omit this attribute to publish macros in this macro library as *global macros*. Domain names: ■ Must be unique. ■ Are case sensitive. ■ Can contain letters, numbers, periods (.), dashes (-) and underscores (_).

Including Macro Domains in Mashup Scripts or Macros

You can use macros in a mashup script or another macro if:

- They are inner macros that are defined within that mashup script or macro.
- They are registered as global macros.
- They are reusable macros and you include the macro domain that defines them in your mashup script or macro.

To include a macro domain in a mashup script or another macro, add <include> directly within <mashup>, <macro> or <operation> and specify the domain. In the Mashup Editor, you can automatically include the appropriate macro domain when you add the macro call to a mashup script. See "Adding Macro Calls Automatically in the Mashup Editor" on page 681 for information.

For example:

```
<mashup name="myMashup"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
     ../xsd/EMMLPrestoSpec.xsd"
    xmlns="http://http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:macro = "http://www.openmashup.org/schemas/v1.0/EMMLMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions">
  <include domain="myConversionMacros"/>
...
```

<include>

A statement to include the macro definitions from a macro domain in a mashup script or another macro. You can only include macros from domains that have been registered in the MashZone NextGen Server that hosts this mashup or macro.

Note: You do not need to include global macros, as they are included automatically.

See also "Including Macro Domains in Mashup Scripts or Macros" on page 646 for more information and examples.

Can Contain	Empty
Allowed In	mashup | operation | macro

Attributes

Name	Required	Description
domain	yes	The name of the macro domain containing EMML macros to include here.

Calling a Macro

Once you have defined macros or included a macro domain in a mashup script or another macro, you call the macro in your mashup or other macro with a *macro reference statement* - an element with the macro name and the *Macro Reference Namespace*.

For a macro named "myMacro", for example, the call to use the macro might look like this:

```
<macro:myMacro domain="myDomain"/>
```

Note: Examples of macro calls in this topic use `macro` as the prefix for the Macro Reference Namespace.

See "Macro Reference Namespace" on page 647, "Macro Domains" on page 648, "Passing Input Parameters to the Macro" on page 648, "Calling Macros From Another Macro" on page 649 and "Receiving Macro Results" on page 649 for information on the options and requirements of macro reference statements. You can also automatically insert macro reference statements in the Mashup Editor. See "Adding Macro Calls Automatically in the Mashup Editor" on page 681 for information.

See also "<macro:custom-macro-name>" on page 650.

Macro Reference Namespace

Custom statements use a separate namespace from EMML because they are unique to your environment. They are not defined in the EMML schema.

When you use macros in a mashup or another macro, the Macro Reference Namespace must be declared in your mashup script or macro. For example:

```
<mashup name="myMashup"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMMLSchema ../schema/EMMLSpec.xsd"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
...
```

```
</mashup>
<macro name="myCustomMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
...
</macro>
```

See "Declaring Namespaces" on page 465 for instructions. See "EMML Namespaces" on page 450 for the current Macro Reference Namespace.

Macro Domains

For macros that are registered in a user-defined domain, you must:

- Supply the name of the domain in the macro call to uniquely identify the macro to invoke. The domain is not required for macros published as *global macros*.
- Add this domain to the mashup or macro using <include>.

  The Mashup Editor automatically adds this <include> statement. See "Adding Macro Calls Automatically in the Mashup Editor" on page 681 for details.

Note: If you omit the domain name in a macro call and the macro name is *not* unique, the MashZone NextGen Server calls the last macro found with that name.

For example:

```
<!-- call global macro -->
<macro:myGlobalMacro input="$myDoc" outputvariable="$macroResult"/>
<!-- call to macro in another domain -->
<include domain="Finance"/>
<macro:myMacro domain="Finance" input="$myDoc" outputvariable="$macroResult"/>
```

Passing Input Parameters to the Macro

In most cases, you also need to pass in input parameters to the macro. Input parameters are defined as attributes on the macro reference statement with a name corresponding to the <input> declarations defined in the macro.

For example:

```
<macro name="myMacro"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
  <input name="email" type="string"/>
  <input name="message" type="string"/>
  ...
</macro>
...
<macro:myMacro email="myTeam@myOrg.com" message="$myMsg"/>
```

As with any EMML attribute, the attributes of the macro reference statement accept references to mashup variables or input parameters.

Receiving Macro Results

If the macro has results, the macro reference statement you use to call the macro also has an implicit *outputvariable* attribute that allows you use to identify the variable in your mashup script or macro that should receive the macro results.

In the example shown below, the results of the macro are accessible inside the macro in the `$macroResult` variable.

```
<macro name="myMacroWithResults"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
  <input name="email" type="string" />
  <input name="message" type="string" />
  <output name="macroResult" type="string"/>
  <assign fromexpr="concat($message,', ',$email)"
    tovariable="$macroResult"/>
</macro>
...
<macro:myMacroWithResults email="myTeam@myOrg.com"
    message="Hello" outputvariable="myResults"/>
<display message="Macro message is " variable="$myResults"/>
...
```

In this example, the macro results are placed in the `$myResults` variable identified in the `outputvariable` attribute when the macro is called.

Calling Macros From Another Macro

Calling a macro from within another macro has the same requirements of calling a macro from a mashup:

- Adding the Macro Reference Namespace to <macro>.
- If the macro belongs to a user-defined domain, including the "Macro Domains" on page 648.
- Using the macro reference statement.

For example:

```
<macro name="reusableMacroWithNestedCall"
    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
  <input name="currentUser" type="string" />
  <output name="macroResult" type="string"/>
  <variable name="currentActivity" type="document"/>
  <variable name="updatedPoints" type="number"/>
 <!-- logic to determine current activity, assign to $currentActivity -->
  <include domain="UserMath"/>
  <macro:calculatePoints domain="UserMath" thisUser="$currentUser"
    activity="$currentActivity" outputvariable="updatedPoints"/>
  <display console="true" message="Updated points are"
    variable="$updatedPoints"/>
...
</macro>
```

<macro:custom-macro-name>

The use of a custom EMML statement defined in a macro of that name. To use macros in a mashup script or macro, you must declare the Macro Reference Namespace and use the namespace prefix with your custom macro name.

See also "Calling a Macro" on page 647 and "EMML Namespaces" on page 450 for more information and examples.

Content Model	Empty. Input parameters to the macro are passed as attributes of the same name.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
domain	Conditional	The domain that this macro belongs to. This is not required for macros in the built-in *global* domain. The domain name ensures that the correct macro is invoked when more than one domain of macros is included in a mashup script. **Note:** If you omit this attribute and the macro name is not unique, the MashZone NextGen Server calls the last macro with this name.
outputvariable		The variable in this mashup script or macro that should receive the results of this macro. This is optional.
any input parameter		Pass input parameters to the macro using attributes of the same name.

Making Mashup Scripts Dynamic

There are two basic types of dynamic behavior for mashup scripts:

- Dynamically changing what EMML statements act on using Dynamic Mashup Syntax to change the values of attributes or make an entire mashup script generic.
- Dynamically changing the flow of mashup logic involves Generating Mashup Scripts Dynamically.

Dynamic Mashup Syntax

Dynamic syntax allows you to parameterize the behavior of individual EMML statements. These are the basic techniques you can use:

- Use Parameters/Variables in Attributes
- Define XPath or Other Syntax Using <template>
- Escape XML Delimiters Inside <template> Expressions
- Use Dynamic Mashup Expressions Inside XPath Expressions
- Create Generic Mashup Scripts

Use Parameters/Variables in Attributes

You can use parameters or variables directly in attributes for any EMML statement. For example:

```
<sort inputvariable="$toSort" outputvariable="$sortResult"
  sortexpr="/records/record" sortkey="key"/>
<invoke service="$servicename" operation="$opname"
  inputvariables="$opinput" outputvariable="service1Result"/>
```

Define XPath or Other Syntax Using <template>

The "<template>" on page 653 declaration can be used to dynamically define XPath expressions, portions of XPath expressions, SQL queries, URLs or any string. The <template> result is assigned to a variable which can be used in other EMML statements or passed as a parameter to other mashup scripts. See "Dynamic XPath or Other Syntax in Mashups" on page 606 for more information and "Escape XML Delimiters Inside <template> Expressions" on page 652.

This example, defines a dynamic join condition, in the `expr` attribute, based on two input parameters. The result is assigned to a variable and then used in a <join> statement:

```
<template expr="{$column1} = {$column2}" outputvariable="$dynamicEqualJoin"/>
<join outputvariable="joinResult" joincondition="$dynamicEqualJoin"/>
```

The next example shows <template> used to build a dynamic SQL query based on input parameters which is then used in a <sql> statement:

```
<input name="offense" type="string" />
<input name="method" type="string" />
<input name="numrows" type="number" default="10" />
...
<variable name="sqlquery" type="string"
  default="select * from crime where 0=0  " />
...
<if condition="$offense != ''">
    <template expr="{$sqlquery} and offense = '{$offense}' "
      outputvariable="sqlquery"/>
  <elseif condition="$method != ''">
    <template expr="{$sqlquery} and method = '{$method}' "
      outputvariable="sqlquery"/>
  </elseif>
</if>
```

```
<template expr="{$sqlquery} limit 0,{$numrows}" outputvariable="sqlquery" />
...
<sql name="DCCrime2008" query="$sqlquery" outputvariable="result"/>
```

Escape XML Delimiters Inside <template> Expressions

When you build expressions with <template> that use XML delimiters natively in the expression, you need to *double-escape* XML delimiters. This is required because <template> expressions are evaluated *twice* to fully resolve expressions.

A common example is building URLs with <template> to use in <directinvoke>. URLs that contain query parameters use the `&` character to separate the parameter/value pairs. In this case, you need to escape the `&` character twice to avoid syntax errors. For example:

```
<template outputvariable="geoCodeEndpoint"
    expr="http://maps.google.com/maps/geo?q={$address}&amp;amp;output=xml"/>
<directinvoke endpoint="$geoCodeEndpoint" method="GET"
    outputvariable="geocode"/>
```

The first evaluation resolves `&` to `&` which is then properly resolved in the second evaluation when the mashup script is run to `&`.

Use Dynamic Mashup Expressions Inside XPath Expressions

Dynamic mashup expressions can be used in any portion of XPath expressions - as long as they are not nested. The first example defines an XPath expression to a node by evaluating the `$selectedItem` variable.

```
{$selectedItem}[matches(description, 'Ruby')
replace({$fix},{$pattern},{$new})
```

The second example passes string parameters to the XPath `replace()` function.

Create Generic Mashup Scripts

You can also make mashup scripts that are generic. Generic mashup scripts define a pattern of processing that can be applied to different mashable information sources or conditions based on input parameters. The flow of logic is the same, but what the logic acts on or how the logic is interpreted is dynamic.

Note: If instead you need the flow of logic within a mashup to change dynamically, see "Generating Mashup Scripts Dynamically" on page 654 for more information.

Using parameter references in <invoke> or <directinvoke> allows generic mashup scripts to work with many different services. This is the most basic usage for a generic mashup script. To make a mashup generic, though, you typically also have to parameterize expressions for other statements using <template>.

This example shows two simple mashup scripts: 1) the generic script that invokes a mashable information source and applies a filter and 2) a mashup that uses the generic mashup as a component to apply to a specific situation.

The Generic Mashup Script

```
<mashup name="FilterOneService" ...>
  <input name="thisService" type="string"/>
  <input name="thisOp" type="string"/>
  <input name="thisOpParams" type="string"/>
  <input name="thisFilter" type="string"/>
  <output name="filteredResult" type="document"/>
  <invoke service="$thisService" operation="$thisOp"
    inputparams="thisOpParams" outputvariable="$initialResult"/>
  <filter inputvariable="$initialResult" filterexpr="$thisFilter"
    outputvariable="$filteredResult"/>
</mashup>
```

The calling mashup defines input parameters using <template> and then invokes the generic mashup script as a component:

A Mashup Using the Generic Mashup Script

```
<mashup name="myMashup" ...>
  <input name="filterBy" type="string"/>
  <variables>
    <variable name="myService" type="string" default="Artima"/>
    <variable name="myOp" type="string" default="getFeed"/>
    <variable name="myOpParams" type="string"/>
  </variables>
  <output name="myResult" type="document"/>
  <template expr="/channel/items/item/{$filterBy}"
    outputvariable="$myFilter"/>
  <invoke service="FilterOneService" operation="Invoke"
    inputparams="myService,myOp,myOpParams,myFilter"
    outputvariable="$myResult"/>
</mashup>
```

<template>

This statement defines an expression with dynamic parameters and assigns that to a variable. This variable can then be referred to dynamically in other mashup script statements that use expressions.

See also "Dynamic Mashup Syntax" on page 651 for more information and examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
expr	yes	The XPath expression to assign to the output variable. Typically, this XPath expression also

Name	Required	Description
		contains one or more mashup expressions that will accept input parameters at runtime. **Note:** The XPath expressions is this attribute are, by definition, evaluated twice to provide the dynamic capabilities of <template>. Because of this, you may need to escape or double-escape XML delimiter characters such as `&`.
outputvariable	yes	The required variable to accept the output of this statement.

Generating Mashup Scripts Dynamically

When the logic of a mashup must change dynamically, you need to create a mashup script on the fly and execute it. You can use the built-in `executeDynamicEMML` macro to do this. This macro is installed with the MashZone NextGen Server and is available for use in any mashup script.

Note: You can use any EMML statement in dynamic mashup scripts, with one *exception*. You cannot invoke another mashup that itself is a dynamic mashup script.

The executeDynamicEMML macro allows you to generate a mashup script dynamically using the <script> statement and JavaScript or using Java classes. See "<script>" on page 523 for more information on using JavaScript. See "Using Java Classes in a <script> Statement" on page 627 for information on using Java classes in a mashup.

<macro:externalDynamicEMML>

Use the following attributes to define the input parameters for this macro and receive the results:

Attribute	Required	Description
script	yes	A string containing the dynamically generated EMML code to execute. Mashup scripts that are generated dynamically have these restrictions: ■ <*input*> statements are not allowed as you cannot pass parameters to the mashup. ■ *Variable access* is limited to the variables defined within the generated code. Generated scripts do not have access to variables in the parent mashup.

Attribute	Required	Description
outputvariable	yes	The name of the variable to receive the results of the generated mashup script. This variable should be the same type as the expected results of the mashup script. Typically this is a document type.

<macro:executeDynamicEMML> Example

In this use case, data for the mashup comes from several different sources as either web services or from databases. But the access method for a particular data source can change at any given time based on configuration. The mashup uses this configuration to generate a mashup that invokes each data source using the appropriate method:

```
....
  <output name="dynamicResult" type="document"/>
  <variables>
    <variable name="serviceConfig" type="document"/>
    <variable name="dynamicScript" type="document"/>
  </variables>
  <invoke service="ServiceConfig" operation="getConfig"
   outputvariable="serviceConfig"/>
  <script type="javascript" inputvariables="$serviceConfig"
     outputvariable="$dynamicScript">
    <![CDATA[
      dynamicScript = "<mashup " +
        "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
        "xsi:schemaLocation=\"http://www.openmashup.org/schemas/v1.0/EMML/ " +
        "../src/schemas/EMMLSpec.xsd\" "+
        "xmlns=\"http://www.openmashup.org/schemas/v1.0/EMML\" " +
        "name=\"myDynamicMashup\"> " +
        "<operation name=\"getFeed\"> " +
        " <output name=\"result\" type=\"document\"/> ";
        if (serviceConfig.serviceA = 'web') {
          dynamicScript = dynamicScript +
            "<invoke service=\"serviceA\" operation=\"getData\" " +
            "outputvariable=\"resultA\" />";
        }
        //more logic to generate mashup script statements
        dynamicScript = dynamicScript +
        " </operation> " +
        " </mashup> ";
    ]]>
  </script>
  <macro:executeDynamicEMML script="$dynamicScript"
    outputvariable="$dynamicResult"/>
...
```

Using the Built-in Mashup Query Web Service

MashZone NextGen has a built-in web service, JEMSServiceHelper, that you can use to query for information about mashups. This topic is a work in progress. More information on using the built-in Mashup Query service is coming soon!

User Metadata Queries

There are two user metadata queries you can make:

- Query for all <user-meta> commands in a specific mashup. You supply the mashup name as a parameter.
- Query for all<user-meta> commands that match a condition (an XPath expression) in a specific mashup. You supply the mashup name and the XPath expression.

Find All User Metadata Requests

Service Name	JEMSServiceHelper
Operation Name	getUserMetadata
Parameters	A single string parameter that is the name of the mashup for the query.

Find Specific User Metadata

Service Name	JEMSServiceHelper
Operation Name	getUserMetadata
Parameters	1. A string parameter that is the name of the mashup for the query. 2. A string parameter that is an XPath expression, relative to <user-meta>, with the condition to select specific user metadata. Typically, this condition matches the <user-meta> name. For example: `@name = "myMeta"`

User Metadata Query Results

The default JSON response for user metatdata queries is in this form:

```
"response":"{
  "serviceMetadata": {
    "user-metas":[
      {"myMeta": "some value"},
        ...
    ]
  }
}
```

In XML format, the result is a document in this form:

```
<serviceMetadata>
  <user-metas>
    <!-- list of all matching commands -->
    <user-meta name="myMeta">some value</user-meta>
    ...
  </user-metas>
</serviceMetadata>
```

MashZone NextGen Metadata Queries

There are two MashZone NextGen metadata queries you can make:

- Query for all <presto:presto-meta> statements in a specific mashup. You supply the mashup name as a parameter.
- Query for all <presto:presto-meta> statements that match a condition (an XPath expression) in a specific mashup. You supply the mashup name and the XPath expression.

Find All MashZone NextGen Metadata as a Mashup

Service Name	JEMSServiceHelper
Operation Name	getPrestoMetadata
Parameters	A single string parameter that is the name of the mashup for the query.

Find Specific MashZone NextGen Metadata with MashZone NextGen Connect APIs

Service Name	JEMSServiceHelper
Operation Name	getPrestoMetadata
Parameters	1. A string parameter that is the name of the mashup for the query. 2. A string parameter that is an XPath expression, relative to <presto:presto-meta>, with the condition to select specific MashZone NextGen metadata. Typically, this condition matches the <user-meta> name. For example: `@name = "cache"`

MashZone NextGen Metadata Query Results

The default JSON response for MashZone NextGen metatdata queries is in this form:

```
"response":"{
  "serviceMetadata": {
    "presto-metas":[
      {"cache": "true"},
        ...
    ]
  }
}
```

In XML format, the result is a document in this form:

```
<serviceMetadata>
  <presto-metas>
    <!-- list of all matching commands -->
    <presto-meta name="cache">true</presto-meta>
  </presto-metas>
</serviceMetadata>
```

Mashable Dependency Queries

You can query to determine what mashables a specific mashup invokes. This query returns information on all <directinvoke> and <invoke> commands in the specified mashup.

Mashable Dependency Requests

Service Name	JEMSServiceHelper
Operation Name	getDependentServices
Parameters	A single string parameter that is the name of the mashup whose dependencies should be queries.

Mashable Dependency Responses

In the default JUMP format, the response is returned as a string in JSON format, such as this:

```
"response":"{
  "services":{
    "direct":{
      "service":[
        {
         "endPoint":"https:\\www.google.com\accounts\ClientLogin",
         "method":"post"
        },
        {
         "endPoint":"$bloggerURL",
         "method":"post"
        }
```

```
      ]
    },
    "presto":{
      "service":{
        "name":"YahooHotJobsRSS",
        "operation":"getFeed"
      }
    }
  }
}
```

The XML format for the response looks something like this:

```
<services>
  <internal>
   <service name="" operation=""/>
   ...
  </internal>
  <external>
    <service endpoint="" method="" />
    ...
  </external>
</services>
```

Internal services are MashZone NextGen mashables. External services are ungoverned mashable information sources that are invoked with the <directinvoke> statement.

Mashup Samples

With theMashZone NextGeninstallation some additional samples are installed.

You can install the samples located in *MashZoneNG-install* /samples/emml using the *MashZoneNG-install* /prestocli/bin/publish-mashups scripts. These scripts will install any of the samples located in *MashZoneNG-install* /samples/emml and its subfolders.

After installation, the samples are shown in the Mashup Editor under **My Mashups** and can be opened directly from there.

Sample Mashup	Description
Annotate annotate.emml	Examples of <annotate> to add nodes and data to existing nodes.
FeedAggregationSample aggfeed.emml	Invokes RSS and Atom Services in parallel using <parallel/> statement. Normalizes all RSS feed types to Atom using feedtype='atom' attribute in directinvoke. After invocation of all feeds in parallel, it subsequently aggregates them using <merge/> and returns a single consolidated feed.

Sample Mashup	Description
Assign to Expressions assignments.emml	Examples of <assign>.
Format as CSV csvmacro.emml	Example of outputing a mashable result in CSV format. Example using FormatAsCSV macro.
Dynamic EMML dynamicemml.emml	Example of how to create a mashup script dynamically and invoke it within another mashup. Examples of <script> and executeDynamicEMML macro.
Dynamic Filter dynamicfilter.emml	Example of invoking a web service and performing a filter based on dynamic filtering criteria. Example of mashup expressions and <filter>.
Dynamic Invoke dynamicinvoke.emml	Example of invoking a mashables that is specified dynamically. This sample invokes the Twitter search web service to find tweets for a term specified as input parameter. Outputs all tweets found. Example of <foreach> and <appendresult>.
Encoding encoding.emml	Examples of setting the character encoding for results.
Scripting POJOs encrypt.emml	Example of handling encryption using <script> and Java classes.
Try/Catch exception-java.emml	Examples of <try>/<catch> for SQL statements, XSLT or other common EMML statements.
Forceably return in loops exception-java-return.emml	Examples of <return> to set the mashup result and forcibly return it based on exceptions.

Sample Mashup	Description
Try/Catch for REST exception-rest.emml	Examples of <try>/<catch> for invocation exceptions from <directinvoke> for REST web services or web feeds.
Try/Catch for SOAP exception-soap.emml	Examples of <try>/<catch> for invocation exceptions from <directinvoke> when the response is a SOAP fault.
Throw Exception exception-throw.emml	Examples using <throw> to throw exceptions from a mashup and a macro.
Fault Handling faulthandling.emml	Example of using error variables from mashable invocations to control processing. Example of <if>, <elseif> and <else>.
Finance News googlefinancenews.emml	Uses web clipping from Google Finance News site to clip and output only specific content. Examples of <template> to have dynamic XPath expressions, <assign>, <foreach> and <appendresult>.
Scripting with Groovy groovy.emml	Example of scripting in a mashup using Groovy.
Data Aggregations using For Each groupby2.emml	Example of grouping using <foreach> and <appendresult>.
Data Aggregation using Group By groupby.emml	Example of single and nested grouping with <group>.
JsonPostPayloadSample jsonPost.emml	Demonstrates doing a HTTP POST of JSON document to Mashup and further manipulation of the JSON document using JSON.parse() and JSON.stringify() methods in javascript.

Sample Mashup	Description
Scripting using Javascript javascript.emml	Example using inline JavaScript in a mashup.
Join join.emml	Demonstrates how two information sources can be joined based on a simple criteria.
Join with Functions joinfunctions.emml	Example of <join> using XPath functions in the join criteria. Also uses <select> to build the joined output.
Join with Select joinselect.emml	Example of <join> using <select> to build the joined output.
Macros macros.emml	Examples of simple macro definitions (custom EMML statements) and how to use them in a mashup.
Union merge.emml	Example of <merge> and <parallel>.
Union of Nodesets mergenodelists.emml	Examples of <merge> when the results are node sets rather than well-formed documents.
ModifyDocument modifydoc.emml	In-place modification of XML DOM element values using <assign mode="replace" .. ./>
newMashup xsDateTimeComparison.emml	Illustrates usage of ISO Date format function.
Invocation Error Handling onerror.emml	Example of the onerror attribute to control mashup processing when invocation errors occur. Example of <if> with <elseif> and <else> subclauses.
Outer Join	Example of an outer join using XQuery.

Sample Mashup	Description
outerjoin.emml	
Parallel parallel.emml	Example of <parallel> invoking two mashables.
HttpMethodsSample rest-methods.emml	Sample invocation of REST API using various HTTP methods - OPTIONS, HEAD, POST, PUT and DELETE
Filter rssfilter.emml	Example of filtering results from a syndicated web feed.
Data Splitter with Select select.emml	Example of <select> statement to filter selected nodes from each item in the result.
SelfJoin selfjoin.emml	Technique for doing SQL-like self join using EMML join statement.
Sort using Functions sort.emml	Examples of <sort>.
User-Defined Functions soundex.emml	Example of using a custom XPath function.
SPARQLQuery sparql.emml	Constructs a SPARL query using Groovy <script/> and invokes a semantic web service to execute it.
SqlResetService sqlreset.emml	Utility Service
SQL sql.emml	Examples of <sql> statement for database queries or stored procedures. Example of <sqlUpdate> and SQL transaction statements.
tryLoadCatch	Uses EMML exception handling mechanism to initialize cache from

Sample Mashup	Description
tryLoadCatch.emml	a datasource on first-invocation and subsequently serves data from this cache.
Unique unique.emml	Example of <filter> to ensure unique results.
Web Clipping webclipping.emml	Example of web clipping from a web site.
While Loop while.emml	Examples of <while>.
xml-names xml-names.emml	
XQuery xquery.emml	Example using XQuery.
XSLT xslt.emml	Example of using XSLT to transform results in a mashup.
Yahoo Finance Stock Quote yahoofinancequote.emml	Example builds a dynamic URL with <template> to web clip information from Yahoo Financial. Uses <assign> and XPath to perform complex clipping and also dynamic mashup expressions within <constructor> to define the result.

Mashup Utilities

You may use the following command-line utilities to publish, update or test mashup scripts or macros outside the Mashup Editor:

- Testing Mashup Scripts from the Command Line
- Publishing Mashup Scripts from a Command Line
- Updating Mashup Scripts from the Command Line
- Publishing and Managing Macros from the Command Line

In addition, you can also enable performance profiling for mashup scripts and use performance logging to tune scripts. See "Profiling Mashup Performance" on page 669 for instructions.

Testing Mashup Scripts from the Command Line

All of the component mashables you invoke in a mashup script must be registered and activated in MashZone NextGento successfully test the mashup.

To test a mashup script from the command line

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.

   Tip: You may wish to put a copy of the mashup script in this directory to simplify paths.

2. Enter the `emml` command in this form:

   `emml -f` *mashup-script-file-name* `-u` *user-name* `-p` *password* `[-D` *params* `- url` *presto-url* `-V -silent]`

   - Use your user name and password for MashZone NextGen.
   - Use the -D option to specify any input parameters for the mashup. Enter parameters as:

     `-D`*param-name* `=`*value* `[-D...]`
   - The URL is optional. It defaults to `http://localhost:`*app-server-port*`/presto/edge/api` where *app-server-port* is the port you defined during installation.

     Use the `-url` option, if you need to test this mashup on a remote server or the default host or port is incorrect.
   - Use the -V option if you only want to validate your mashup script. This option does not actually execute the mashup.
   - Use the -silent option to skip the output from any <display> statements.

The mashup script is processed. Any debugging messages in <display> statements will display in the command window in addition to being logged.

Note: Occasionally you may see an error message `Invalid parameter Q` when you test mashup scripts. This error has no effect on processing and can safely be ignored.

You can also receive errors that must be corrected if the names of the mashables that are invokved within the mashup script are not valid for the MashZone NextGen Server used in the test.

Publishing Mashup Scripts from a Command Line

Once you are satisfied with the functionality of your mashup script, you can publish it as a mashup in MashZone NextGen from a command line.

To publish a mashup script from a command line

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.

   You may wish to put a copy of the mashup script in this directory to simplify paths.

2. Enter the `emmlpub` command in this form:

   `emmlpub -f` *mashup-script-filename* `-u` *user-name* `-p` *password* `[-s` *mashup-name* `-desc` *description* `-t` *tag* `[...] -app` *category* `[...] -P` *provider* `-url` *presto-url* `]`

   - Use your user name and password for MashZone NextGen.

     Note: The credentials you use with this command identify the owner of the mashup created from this mashup script. Only the owner or MashZone NextGen administrators can modify or update the mashup.

   - The URL is optional. It defaults to `http://localhost:`*app-server-port*`/presto/edge/api` where *app-server-port* is the port defined when you installed MashZone NextGen.

     If you need to register this mashup to a remote server or the default host or port is incorrect, use `-url http://`*app-server:port* `/presto/edge/api` with the appropriate application server address and port.

   - The mashup name, description, provider, tags and category are optional information. If the values for these options contain spaces, you must enclose the values in quotes.

     If you omit the mashup name, MashZone NextGen uses the name in the `name` attribute on <mashup> in the mashup script. You can change this name in MashZone NextGen Hub at any time. The MashZone NextGen Server also assigns a permanent ID to the mashup.

     MashZone NextGen removes or changes some characters in resource names to create resource IDs. Resource IDs can contain characters from the character sets supported by theMashZone NextGenRepository and numbers plus the following symbols: `_ ~ -`. Resource IDs cannot contain spaces.

   - You can assign multiple tags, each beginning with the -t option.

   - You can assign a category with the -app option.

     Note: You cannot use category names that contain spaces with this command. Simply omit this option and add the category name in MashZone NextGen Hub.

Once the script is published, you see a confirmation.

Updating Mashup Scripts from the Command Line

You update metadata, such as tags or the description, of a mashup in MashZone NextGen Hub just as you would with any mashable information source that you had published.

To update the behavior of mashup scripts that you have published in MashZone NextGen, you must be the owner of the mashup (the user that first published this mashup script) or have MashZone NextGen administrator privileges. You can update the mashup in Mashup Editor or use the command line.

To update a mashup from the command line

1. Edit the EMML file for the mashup, as needed. See "Creating a Mashup Script with EMML" on page 443 for information on EMML options.
2. Turn the mashup off in MashZone NextGen Hub.
3. Test the new mashup script.

   See "Testing Mashup Scripts from the Command Line" on page 665 for instructions.
4. Open a command window and move to the *presto-instalation* /prestocli/bin folder.

   You may wish to put a copy of the updated mashup script in this directory to simplify paths.
5. Enter the `emmlupd` command in this form:

   `emmlupd -f` *mashup-script-filename* `-u` *user-name* `-p` *password* `[-url` *remote-presto-url* `-s` *mashup-ID* `]`

   - Use your user name and password for MashZone NextGen.
   - The URL is optional. It defaults to `http://localhost:`*app-server-port*`/presto/edge/api` where *app-server-port* is the port defined when you installed MashZone NextGen.

     If you need to update this mashup on a remote server or the default host or port is incorrect, use `-url http://`*app-server:port* `/presto/edge/api` with the appropriate application server address and port.
   - The mashup ID is optional. If you omit this option, MashZone NextGen uses the name for the mashup supplied in the <mashup> tag in the mashup script to identify the mashup to update.

Tip: Mashup owners and MashZone NextGen administrators can change mashup names in MashZone NextGen Hub. It is a good practice to provide the mashup ID to properly identify the mashup you want to update.

6. Turn the mashup back on in MashZone NextGen Hub.

Publishing and Managing Macros from the Command Line

You can publish individual EMML macros in the Mashup Editor or from the command line. You can also publish macros in bulk, list macros or macro domains or update, rename or delete macros from the command line.

To perform any of these actions from the command line, macros *must* be defined in macro libraries. See "Create a Reusable Macro Library" on page 645 for more information.

To publish macros from a command line

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter the `macropub` command in this form:

   `macropub -u` *user-name* `-p` *password* `[-url` *presto-url* `]` *action-options*

   where the options for specific actions can be:

Action	
List macro domains	`-l` Lists the names of all macro domains.
List macros	`-l (-g | -dm` *domain-name* `)` Lists the names of all macros in the global domain (-g) or in the domain (-dm) specified.
Publish a macro library	`-lib -f` *macro-library* `[-g]` Publishes all the macros defined in the specific macro library file. If the -g option is set, these are published to the global domain. Otherwise, they are published to the domain specified in the macro library file.
Republish a macro library	`-lib -repub -f` *macro-library* `[-g]` Republishes all the macros defined in the specific macro library file. If the -g option is set, these are published to the global domain. Otherwise, they are published to the domain specified in the macro library file.
Publish a single macro	`-a -f` *macro-library* `-n` *macro-name* `(-g | -dm` *domain-name* `)`

Action	
	Publishes the macro specified by name from the macro library specified to either the global domain (-g) or to the domain (-dm) specified.
Update a macro	`-m -f` *macro-library* `-n` *macro-name* `(-g | -dm` *domain-name* `)` Modifies the macro specified by name from the macro library specified to either the global domain (-g) or to the domain (-dm) specified.
Rename a macro	`-r -n` *new-macro-name* `-o` *existing-macro-name* `(-g | -dm` *domain-name* `)` Changes the name of an existing macro in either the global (-g) domain or in the domain (-dm) specified.
Delete a macro	`-d -n` *macro-name* `(-g | -dm` *domain-name* `)` Deletes the specified macro from the global (-g) domain or the specified (-dm) domain.

The URL is optional. It defaults to `http://localhost:`*app-server-port*`/presto/edge/api` where *app-server-port* is the port you defined during installation.

Use the `-url` option, if you need to publish macros to a remote server or the default host or port is incorrect.

With the file option (-f), you may wish to put a copy of the macro library in the same directory as the command to simplify paths.

Profiling Mashup Performance

You can enable or disable performance profiling for mashup scripts. By default, profiling is disabled but statistics are configured to be sent to the console.

Performance profiling tracks execution time within the MashZone NextGen Server for each EMML statement or declaration. This does not include network latency or any other processing time within the MashZone NextGen Server, such as authentication or response formatting.

See "Configuring Profiling Logging" on page 670 for instructions to enable profiling logging. See "Mashup Profile Logs" on page 670 for information and examples on how profiling is logged.

Configuring Profiling Logging

By default, profiling statistics are disabled but are configured to be sent to the console. To enable analysis, you must enable profiling in the Admin Console.

Mashup Profile Logs

The log file is an XML file that logs the following information for each mashup script that is invoked:

```
<script name="mashup-name">
  <begin time="2008 Mar 20 21:16:36:781 PDT"/>
  <command name="command-name" millis="milliseconds" line="line-naume"/>
...
  <end time="2008 Mar 20 21:16:37:125 PDT"/>
  <total millis="34"/>
</script>
```

EMML Reference

This topic lists EMML declarations and statements with links to their syntax topics.

<annotate>	Adds attributes or children elements to a selected node of a variable and assigns values using an XPath expression. This is typically used to add metadata to the mashup result or any intermediate variable.
<appendresult>	Appends a well-formed structure of nodes to the result of this mashup or to any intermediate variable. This is typically used in repeating loops to handle results with repeated sections.
<assert>	Defines a logical assertion comparing variable, literals, counts or element depths. Assertions must be true to continue mashup processing.
<assign>	Copies, and optionally transforms, one variable or variable fragment to another variable. This uses XPath 2.0 expressions to identify the source nodes to copy and optionally applies XPath functions to transform the result. This statement can also assign literal values.

<break>	To explicitly break out of looping statements.
<constructor>	Creates a well-formed structure for the result of this mashup or for any intermediate variable. You define the nodes of the result and use mashup expressions to define the data from a variable to fill these nodes.
<datasource>	Defines connection information to one database for use with direct SQL commands. See Declaring Data Sources.
<directinvoke>	Invokes a publically-accessible web service or web site that is not governed by MashZone NextGen.Only REST, SOAP or syndication (RSS or Atom) services are currently supported.
<display>	Sends a message to the log and console, optionally including the value of a variable or portion of a variable defined in an XPath 2.0 expression
<emml-meta>	Defines metadata for the mashup specific to EMML. See Adding Metadata to Mashups.
<filter>	Filters a set of nodes in a variable based on a filter expression in XPath 2.0.
<for>	Processes a loop of mashup statements based on a count.
<foreach>	Processes a loop of mashup statements based on a set of nodes from a variable. The set of nodes is an XPath 2.0 'sequence' defined by an expression. Loops may be processed sequentially (the default) or concurrently.
<group>	Sorts, and optional filters, a set of nodes in a variable and constructs a result document based on those groups. This command supports multiple levels of grouping and

	calculations or other transformations on groups.
<if>	Handles if-elseif-else processing for a mashup script based on a condition defined in XPath 2.0. You can include any mashup statement in any section of <if>.
<include>	To make reusable macros from a macro library accessible in a specific mashup script or a specific macro. See also Including Macro Domains in Mashup Scripts or Macros
<input>	Defines a parameter to use as the input to a mashup or macro. See Declaring Mashup and Macro Variables and Parameters.
<invoke>	Invokes the operation for a registered MashZone NextGen mashable information source.
<join>	Joins the results of two or more mashable information sourcess based on a join condition. This statement works much like a database join, where the results may be disparate but must have key nodes that determine how data is joined. You can also define the structure and nodes to include in the result of the join.
<loadfrom>	Loads data to a variable from the MashZone NextGen Analytics In-Memory Stores. This is typically used for very large datasets. **Note:** This is an extension statement for the Real-Time Analytics Query Language and is only available if you have installed the MashZone NextGen Analytics.
<macro>	Allows you to define custom mashup statements for use in one or many mashups. Macros can also be used to define custom blocks in Wires. You can use virtually mashup statement in a macro. Effective with this release you can

	call macros within a macro and define macros within a macro.
<macros>	Root node for macro libraries that contain macro definitions for use in any mashup. See also Creating and Registering Macros.
<macro:custom-macro-name>	To use a custom mashup statement. See also Calling a Macro.
<mashup>	The root element for a mashup script. See also EMML Namespaces.
<merge>	Merges the results of two or more mashables or mashups that have homogenous result models. This statement works much like a database union. The results of all services must have identical structures.
<output>	Defines the name and type for the result of this mashup. See Declaring Mashup and Macro Variables and Parameters.
<operation>	An optional name for the mashup operation.
<parallel>	Processes each <sequence> child concurrently as separate parallel flows. Statements within <sequence> are processed sequentially. <sequence> can contain variable declarations and other mashup statements.
<presto:beginTransaction>	A MashZone NextGen extension statement to begin a transaction specifically for invocations of Database mashables. See Database Mashable Transactions.
<presto:commitTransaction>	A MashZone NextGen extension statement to commit a transaction specifically for invocations of Database mashables. See Database Mashable Transactions.
<presto:macro-meta>	A MashZone NextGen extension statement to provide metadata for macros. This metadata determines whether a macro also defines a custom block in Wires. If so, the metadata

	can also provide configuration for the block properties for the custom Wires block. See Adding Custom Blocks to MashZone NextGen Wires Using Macros for more information.
<presto:presto-meta>	A MashZone NextGen extension statement that allows you to set built-in processing flags for the mashup using MashZone NextGen metadata. See Adding Metadata to Mashups.
<presto:rollbackTransaction>	A MashZone NextGen extension statement to roll back a transaction specifically for invocations of Database mashables. See Database Mashable Transactions.
<raql>	Executes a RAQL query to perform calculations or otherwise transform a very large dataset. **Note:** This is an extension statement for the Real-Time Analytics Query Language and is only available if you have installed the MashZone NextGen Analytics.
<return>	Forcibly stops further mashup or macro processing and returns the current value of <output> as the mashup or macro result.
<script>	Calls a user-defined script to execute at runtime at the specified location in mashup processing. You can include scripting code directly or point to an external file on the local server.
<select>	Creates a structure for the mashup result or for an intermediate variable with only selected nodes from a repeating set of items.
<snapshot>	Defines a variable and populates it with a set of snapshots from the MashZone NextGen Snapshot Repository based on the specified query. **Note:** This is an extension statement for the Real-Time Analytics Query Language and is

	only available if you have installed the MashZone NextGen Analytics.
<sort>	Sorts a set of nodes in a variable based on sort keys and a sorting expression in XPath 2.0.
<sql>	Executes SQL queries directly against a datasource.
<sqlBeginTransaction>	Begins a transaction for SQL commands to a single datasource. See SQL Transactions.
<sqlcall>	Executes a stored procedure against a datasource.
<sqlCommit>	Commits a transaction for SQL commands to a single datasource. See SQL Transactions.
<sqlRollback>	Rolls back a transaction for SQL commands to a single datasource. See SQL Transactions.
<sqlUpdate>	Executes any other SQL statement against a datasource.
<storeto>	Stores data from a variable, typically with a very large dataset, in the MashZone NextGen Analytics In-Memory Stores. **Note:** This is an extension statement for the Real-Time Analytics Query Language and is only available if you have installed the MashZone NextGen Analytics.
<template>	Defines XPath expressions dynamically that can be passed into generic mashup scripts. See Dynamic Mashup Syntax.
<throw>	Throws the exception defined in the body of this statement. If this exception is not caught by a subsequent <catch> block, this also forcibly stops further mashup or macro processing.
<try>	Handles try-catch processing for mashups or macros, catching exceptions thrown from

	any EMML statements in the <try> block. Depending on the <catch> block that matches an exception, this may: ■ Allow processing to continue. ■ Throw a new exception. ■ Forcibly stop further processing, returning the current result. See Handling or Throwing Exceptions for more information.
<user-meta>	Allows you to define your own metadata for a mashup or macro. See Adding Metadata to Mashups.
<variables>	Contains one or more definitions of variables to hold the input, output or intermediate data for any mashup statement . See Declaring Mashup and Macro Variables and Parameters.
<while>	Processes a loop of any mashup statements as long as the specified condition is true. The condition is defined in an XPath 2.0 expression.
xmlns on <mashup> or <macro>	Declares a namespace used in a mashup script or macro. See Declaring Namespaces.
<xslt>	Transforms an document-type input variable using an XSLT 2.0 stylesheet.

In addition, the following common content model groups are used in many EMML statements: Declarations Group, Macroincludes Group, Statements Group and Variables Group.

Declarations Group

Declarations define the parameters and meta data used within a mashup. This includes input and the result of the mashup. Meta data may be user- or system-defined. See also the Variables Group for additional declarations.

Valid content for this group is any choice of the following:

- emml-meta
- input

- output
- user-meta

Macroincludes Group

Macros define custom statements for use in a specific mashup or in any mashup hosted by a given MashZone NextGen Server. They are 'mini-mashups.'

Include statements allow you to include macro definitions from a macro library in a mashup or in another macro library.

Valid content for this group is any choice of the following:

- include
- macro

Statements Group

Statements are the actions that the mashup performs.

Valid content for this group is any choice of the following:

- annotate
- appendresult
- assign
- assert
- constructor
- directinvoke
- display
- filter
- for
- foreach
- group
- invoke
- if
- join
- merge
- parallel
- return

- script
- select
- sort
- sql
- sqlcall
- sqlBeginTransaction
- sqlCommit
- sqlRollback
- sqlUpdate
- template
- throw
- try
- while
- xslt
- If the MashZone NextGen Analytics is installed:
  - load
  - raql
  - snapshot
  - store

Variables Group

This group contains additional declarations that define variables and datasources that are used within a mashup. See the Declarations Group for other types of declarations.

Valid content for this group is any choice of the following:

- datasource
- variables
- variable

7 Working in the Mashup Editor

- Editing Short Cuts and Tips 680
- Automatically Generating <invoke> Statements in Mashup Editor 681
- Adding Macro Calls Automatically in the Mashup Editor 681
- Editing a Mashup in the Mashup Editor 682
- Test Mashup Scripts or Macros in Mashup Editor 683
- Publish a Mashup Script in the Mashup Editor 684

Since EMML is an XML language, developers can use any XML-aware editor to write mashups and macros. The Mashup Editor in MashZone NextGen Hub allows you to easily work with EMML and MashZone NextGen mashables, mashups and macros to create, edit, test and publish mashup scripts and macros.

You can use the Mashup Editor to:

- Create mashups or macros using EMML. See "Creating a Mashup Script with EMML" on page 443, and "<macro>" on page 638 for details.

  You can also automatically generate the EMML statements needed to invoke MashZone NextGen mashable information sources, mashups and macros using Mashup Editor menus. See "Automatically Generating <invoke> Statements in Mashup Editor" on page 681 for details.

  Note: Mashup Editor saves mashups and macros directly to the MashZone NextGen Repository. This makes mashups and macros instantly available, but can complicate common development practices such as using source control software.

- Easily edit a mashup you have started in Wires and extend the mashup with EMML. View and copy the EMML code in Wires and paste it into the Mashup Editor. Then edit, as needed.

  Note: If you update the EMML code of mashups created in Wires, they can no longer be edited in Wires.

- Run mashup scripts or macros to see their results or errors. See "Test Mashup Scripts or Macros in Mashup Editor" on page 683 for instructions.

  You can also review performance statistics for mashup scripts or macros to help identify potential performance problems.

- View, test and copy the mashup samples provided in MashZone NextGen. See "Mashup Samples" on page 659 for more information.

For a quick look-up of EMML syntax, see the "EMML Reference" on page 670. For editing tips in the Mashup Editor, see "Editing Short Cuts and Tips" on page 680.

Editing Short Cuts and Tips

The Mashup Editor editing panel supports many common key short-cuts available in text editors including:

Action	Short Cuts in Windows and Linux	Short Cuts in OS/X
Undo	Ctrl + Z	Command + Z

Action	Short Cuts in Windows and Linux	Short Cuts in OS/X
Redo	Ctrl + Y	Command + Z
Cut	Ctrl + X	Command + X
Copy	Ctrl + C	Command + C
Paste	Ctrl + V	Command + V
Reformat	Ctrl + Alt + F	Command + Alt + F

You can also use the **Font Size** menu to increase the default size of text in the Mashup Editor.

Automatically Generating <invoke> Statements in Mashup Editor

You can use the **Mashables** and **Mashups** menu in the Mashup Editor to automatically generate and popultate <invoke> statements for MashZone NextGen mashables or mashups.

1. Move the cursor to the line where you want to invoke a mashable information source or mashup in the mashup script or macro you are editing.
2. Expand the **Mashables**or **Mashups**menu, if needed, and click on the mashable or mashup to invoke.

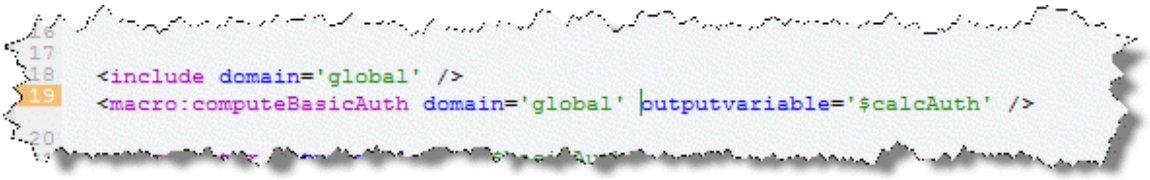
   The entry expands to list all of the operations for that mashup or mashable.
3. Click on the operation to generate the correct statements.

Adding Macro Calls Automatically in the Mashup Editor

To use a macro in a mashup script or another macro, you add an <include> statement for the domain the macro belongs to and add the custom statement for that macro. You can automatically generate both the <include> and custom statement for a macro in Mashup Editor.

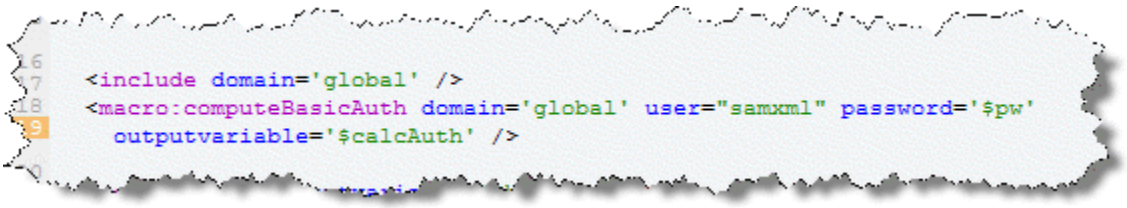
1. Open the mashup script in theMashup Editorand move the cursor to the line where you want to call the macro.
2. Expand the **Macros**menu and click on the macro you want to call.

This inserts both an <include> statement for the domain that the macro belongs to and the custom statement to invoke the macro itself. For example:



3. If the macro has input parameters, add attributes for each input you want to use with this call to the macro.

   You can enter literal values, XPath expressions or references to variables from this mashup for the input parameters. For example:



4. In the`outputvariable`attribute, enter a reference to the variable that should receive the results of the macro.

Editing a Mashup in the Mashup Editor

You must have appropriate MashZone NextGen permissions to edit a mashup in the Mashup Editor.

1. Select **Mashups >** Mashup Editor in the MashZone NextGen Hub main menu to open the Mashup Editor.
2. Click **Open**.
3. Select the mashup you want from the list and click **Open**.

   Use the **Filter** to quickly find the mashup by name.
4. Edit the EMML code as needed, using the Actions, Macros and Mashables menus to insert statements if desired.
5. Run the mashup to test your changes. See "Test Mashup Scripts or Macros in Mashup Editor" on page 683 for more information.
6. Click **Save**, update the information for this mashup if needed and click **Save** to save these changes.

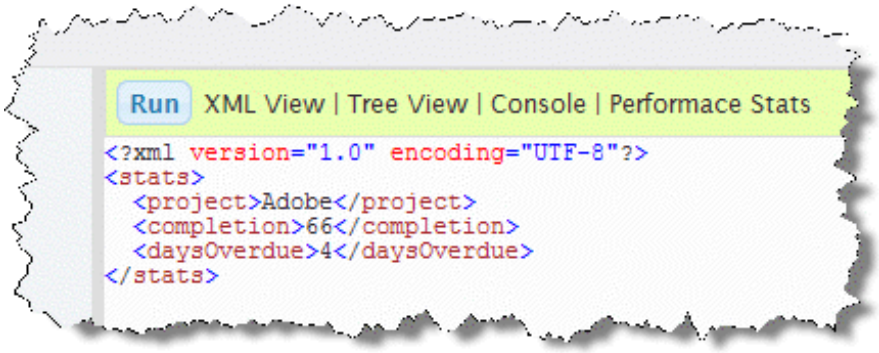
Test Mashup Scripts or Macros in Mashup Editor

You can run mashups or macros in the Mashup Editor to test them. Simply open a mashup or macro and click **Run**.

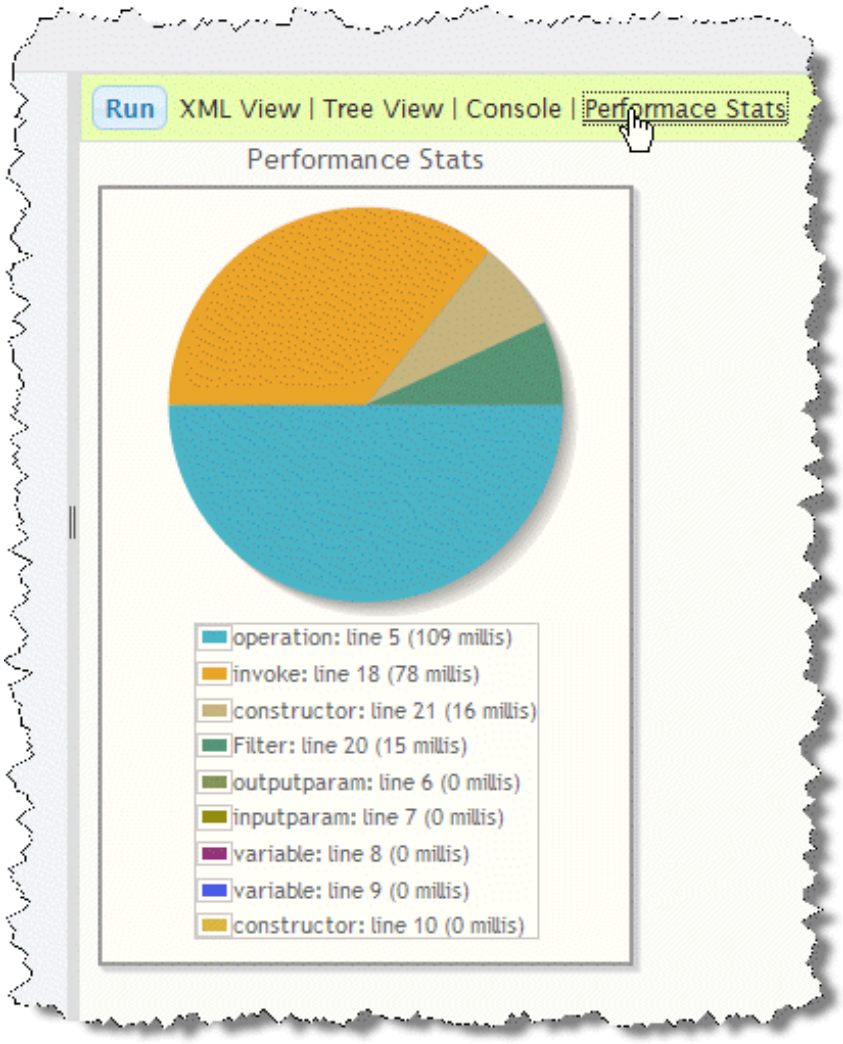
Tip: If the mashup script or macro has input parameters, you may need to set default values in order to run the mashup or macro successfully.

For simple inputs, use the `default` attribute. For complex inputs, define default values as literal XML within the <input> statement.

View the results of the test in the **Output** panel as XML or as a tree. You can also see the output from <display> or <assert> statements or any errors in the **Console** panel.



To see the performance statistics of the test by statement, click **Performance Statistics.**



Publish a Mashup Script in the Mashup Editor

Publishing a mashup script turns the mashup on and allows other users to work with it. You can publish a mashup anytime you save it from the Mashup Editor by setting the **Publish** option when you save it.

If you save a mashup and clear the **Publish** option, you can edit and test it. Other users can see the mashup, but it is turned off.

8 MashZone NextGen Analytics

- What is MashZone NextGen Analytics? 686
- Getting Started with MashZone NextGen Analytics 689
- RAQL Queries 704
- Working with MashZone NextGen Analytics In-Memory Stores 857
- RAQL Extension to EMML Statements for Mashups 876
- RAQL Samples 887

MashZone NextGen Analytics provides a simple method to work with 'big data' whether that is event-driven, real-time, historical or transactional data to produce useful insights and intelligence delivered in visual apps that go anywhere your users need. You get the power and flexibility of mashups, a new query language to easily support your analytic needs, and the support for fast access to data, using streaming and memory management for large datasets.

Note: License requirements for MashZone NextGen Analytics in previous releases are no longer applicable.

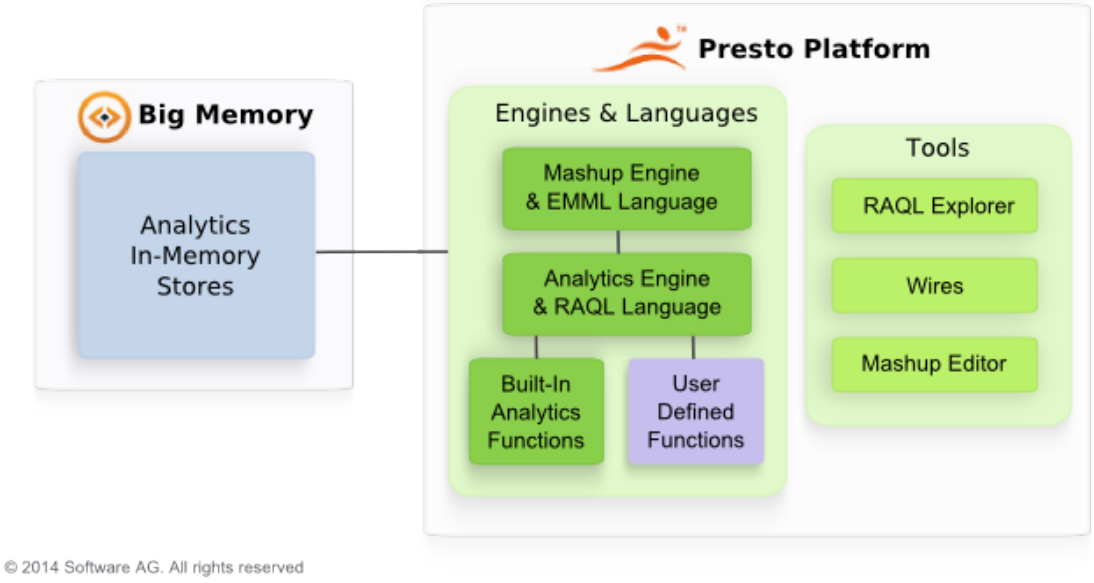
For a short introduction, see "What is MashZone NextGen Analytics?" on page 686. Or if you'd rather dip your toes in first, try Getting Started with MashZone NextGen Analytics.

What is MashZone NextGen Analytics?

Business intelligence daily seems to encompass more data from very different sources: social media for your customers, increasing volumes of real-time data for monitoring operations or managing risk, plus all the traditional historical and transactional data of your organization. Managing this 'fire hose' of data and making sense of it requires:

- Tools to easily combine, explore and analyze data.
- An analytics toolbox to grapple with data complexity.
- Many, flexible ways to visualize insights so they tell a clear, evocative story.

MashZone NextGen Analytics leverages many of the existing features in MashZone NextGen to do this and adds features to let you solve this problem:



- **MashZone NextGen Analytics In-Memory Stores**: to handle 'big data' with good performance using streaming access and Terracotta BigMemory. By default, MashZone NextGen includes limited use of both local *heap* and *off-heap* memory through BigMemory for the MashZone NextGen Analytics In-Memory Stores.

  Adding BigMemory licenses provides straight-forward, highly extensible, fast in-memory data management in either a separate server and host or in an array of distributed servers using an easily-extended architecture.

  Streaming large datasets also removes other performance bottlenecks, such as keeping the entire dataset in MashZone NextGen memory with a DOM to support XPath queries. Data is streamed in chunks (partitions) instead.

- **The Real-Time Analytics Query Language(RAQL) and Analytics Engine**: to query large datasets in a simple but powerful way, without the overhead of a DOM, and easily apply analytic functions. Datasets use a flat, table-like structure to support queries.

  RAQL (pronounced rä·`kel) is a SQL-like query language that provides performant access and handles streaming datasets. It includes a set of built-in analytics functions and a straight-forward way to provide your own user-defined functions to meet new or unique analysis needs.

  You use RAQL queries within mashups, thus leveraging the power and flexibility of mashups. MashZone NextGen Analytics has extended EMML, the language for mashups, to provide streaming access to common sources for datasets as well as store datasets in or load dataset from In-Memory Stores.

- **Mashup Tools** : to create RAQL queries for mashups in Wires, using simple graphic modelling and drag and drop blocks. Or use the full power of EMML + RAQL in the Mashup Editor.

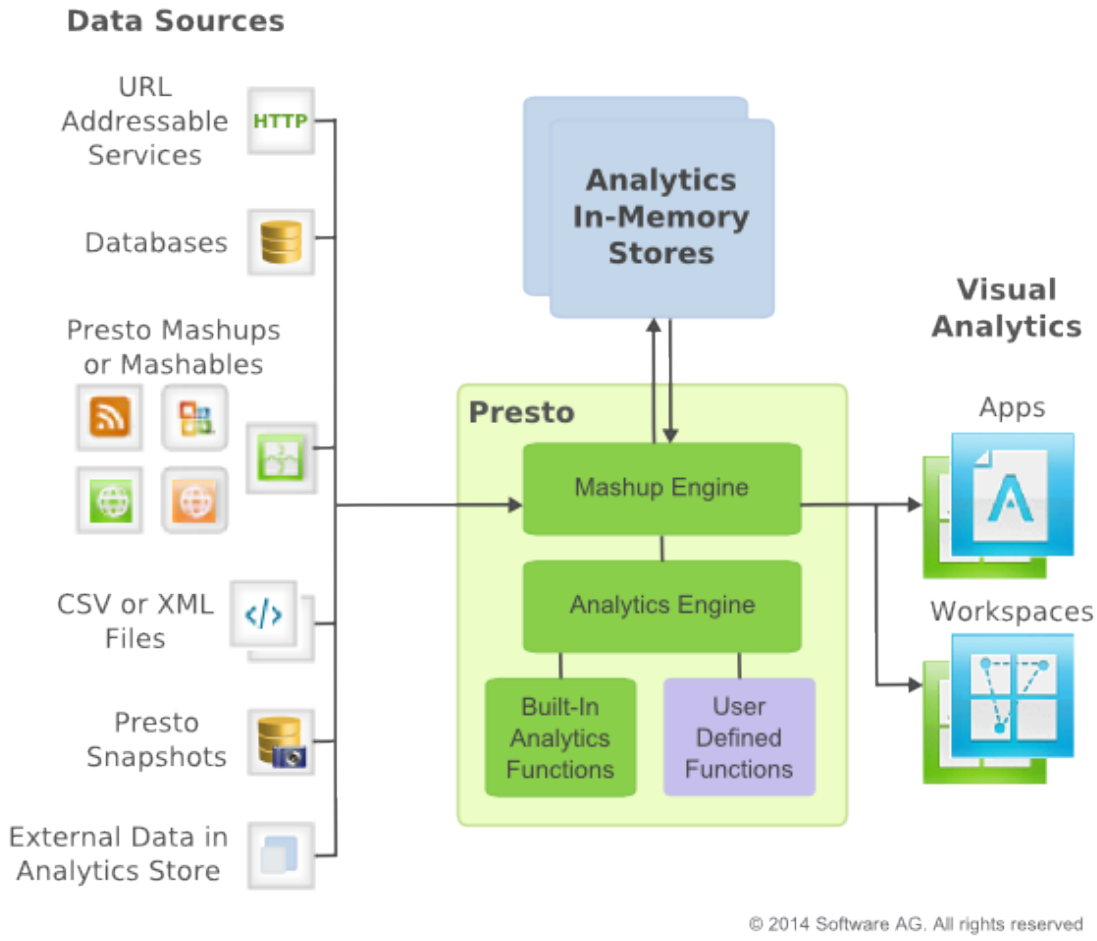
  Once you have the mashup and RAQL queries you need to work with a dataset, simply add views and create apps or workspace apps using MashZone NextGen's point-and-click wizards. Use the built-in MashZone NextGen views or developers can create pluggable views and add them to MashZone NextGen to meet your specific visuallization needs.

MashZone NextGen Analytics: Features and Flow

MashZone NextGen Analytics and RAQL give you access to datasets from a variety of sources, including:

- Databases
- CSV or XML files
- Services addressable by URL
- MashZone NextGen mashables or other mashups
- MashZone NextGen snapshots of results for mashables or mashups
- Datasets that MashZone NextGen or other systems have already loaded in MashZone NextGen Analytics In-Memory Stores

Users work with the final visual analytics that are published as apps or workspace apps. These apps are based on the mashups with RAQL queries and the views you choose to include in the app.



When users find and use these apps, the associated mashup is processed in MashZone NextGen Server using the EMML Engine, the Analytics Engine and any analytics functions defined in RAQL queries. They may work with datasets in MashZone NextGen Analytics In-Memory Stores or directly access datasets from any supported data source.

Next Steps

For your next steps, see "Getting Started with MashZone NextGen Analytics" on page 689, "RAQL Queries" on page 704 and "Working with MashZone NextGen Analytics In-Memory Stores" on page 857.

Getting Started with MashZone NextGen Analytics

This topic presents basic examples to help you get comfortable with the features of MashZone NextGen Analytics and the Real-Time Analytics Query Language (RAQL).

Additional query techniques for RAQL are discussed in RAQL Queries and Working with MashZone NextGen Analytics In-Memory Stores.

For some simple examples of mashups and information on the Mashup Samples project, see "RAQL Samples" on page 887.

About the Real-Time Analytics Query Language Examples

Many of the example datasets used in this topic or other topics illustrating RAQL are available as both:

- Files in the *web-apps-home* /presto/WEB-INF/classes folder in the MashZone NextGen Server.
- Hosted resources at `http://mdc.jackbe.com/prestodocs/data/`*file-name* .

In a few cases, such as examples for snapshots, you must provide some initial configuration or perform some steps in MashZone NextGen Hub to make the datasets used in the example available.

Note: The example datasets used in this topic do not necessarily represent actual load or throughput requirements. Base memory settings for MashZone NextGen may require tuning to provide adequate performance for actual loads. For more information, see "About BigMemory and the MashZone NextGen Analytics In-Memory Stores" on page 861.

First let's take a look at a simple RAQL query.

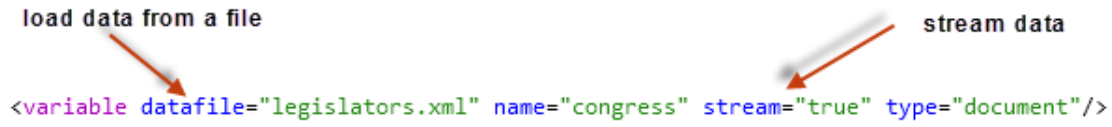
A Basic RAQL Query

You use mashups in MashZone NextGen to access and work with small or large datasets. To work with large datasets, mashups use EMML extension statements specifically designed for MashZone NextGen Analytics and the RAQL query language.

Tip: If you are not yet familiar with creating mashups or EMML, try the EMML in 15 Minutes tutorial for a brief introduction.

To work with large datasets, a mashup must:

1. Load the dataset and give it a name.



This example loads the dataset directly from a file. You can also load data from URLs, databases, snapshots, any MashZone NextGen mashable or mashup or from datasets already stored in one of the MashZone NextGen Analytics In-Memory Stores.

This example also ensures that the data is loaded using a stream. Streaming helps performance and also prevents the dataset from being treated as a *document*. Only RAQL queries or other RAQL extensions to EMML can work with dataset streams. This also affects what data the mashup can return in results. We will cover this in more detail a little later in Getting Started.

2. Run a RAQL query to filter, sort and analyze the dataset stream.

```
<raql outputvariable="result">
  select firstname, lastname, state, chamber from congress
</raql>
```

The `<raql>` extension element runs the query. The syntax for RAQL looks very similar to SQL. The `congress` variable holds the dataset stream and the RAQL engine automatically determines which elements from this XML file represent the *rows* of this dataset.

> **Note:** With XML datasets, you may add path information to override or clarify this auto-detection. See "Dataset Paths, Names and Datatypes" on page 721 for more information.

The complete mashup for this example is:

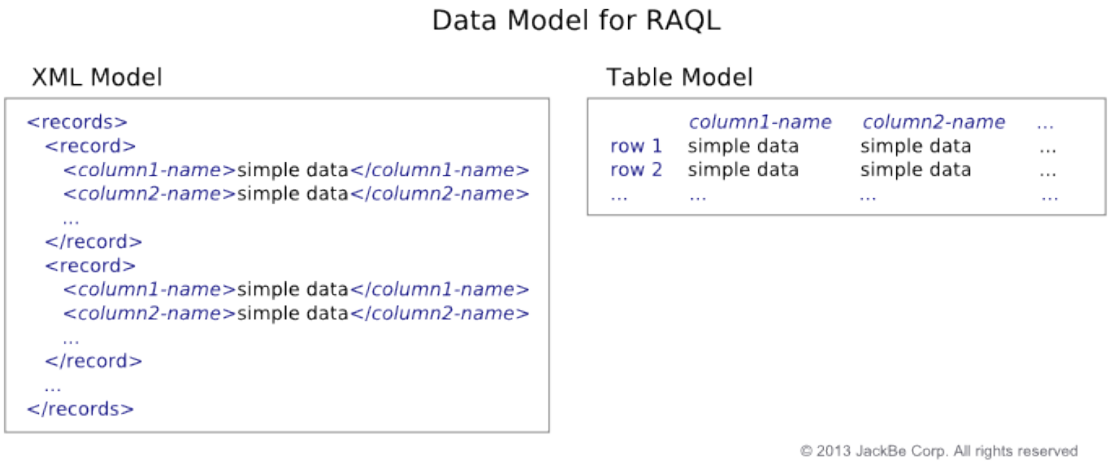
```
<mashup name="BasicQuery"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openmashup
  <operation name="Invoke">
    <presto:presto-meta name="created-using">RUI</presto:presto-meta>
    <presto:presto-meta name="alias">congress</presto:presto-meta>
    <presto:presto-meta name="filename"> legislators.xml
    </presto:presto-meta>
    <output name="result" type="document"/>
    <variable datafile="legislators.xml" name="congress" stream="true"
      subType="xml" type="document"/>
    <raql outputvariable="result">
      select firstname, lastname, state, chamber from congress
    </raql>
  </operation>
</mashup>
```

Structure, Format and Access to Datasets with RAQL

Since data in large datasets may come from many different sources, the data must be in a format and structure that is supported by RAQL. Valid data formats include:

- Comma-separated-values (CSV).
- XML
- JDBC Result Sets (from a database).
- Java objects stored in In-Memory Stores by external systems.

The structure of the dataset must also be flat, like a database table, containing two or more rows (records). Each row must contain at least one column with simple data. Each column must have a unique name.



JDBC result sets from database queries are always in the correct structure. For CSV, XML and Java objects, however, this imposes some specific restrictions. See "Supported Data Formats for RAQL" on page 771 for details.

You can load large datasets in a mashup from:

- *Files*: using `<variable>`. Files must be in the classpath for the MashZone NextGen Server.
- *URLs*: using `<directinvoke>` . This can be a web service or any hosted resource with an appropriate data format. This supports both GET and POST methods.
- *Any MashZone NextGenmashable or mashup*: using `<invoke>`.

> **Note:** Currently, `<invoke>` does *not* support streaming access to data. Results from `<invoke>` are documents and thus may have performance issues for large datasets.
>
> You can, however, run RAQL queries against result documents as long as some portion of the document matches the required table-like structure.

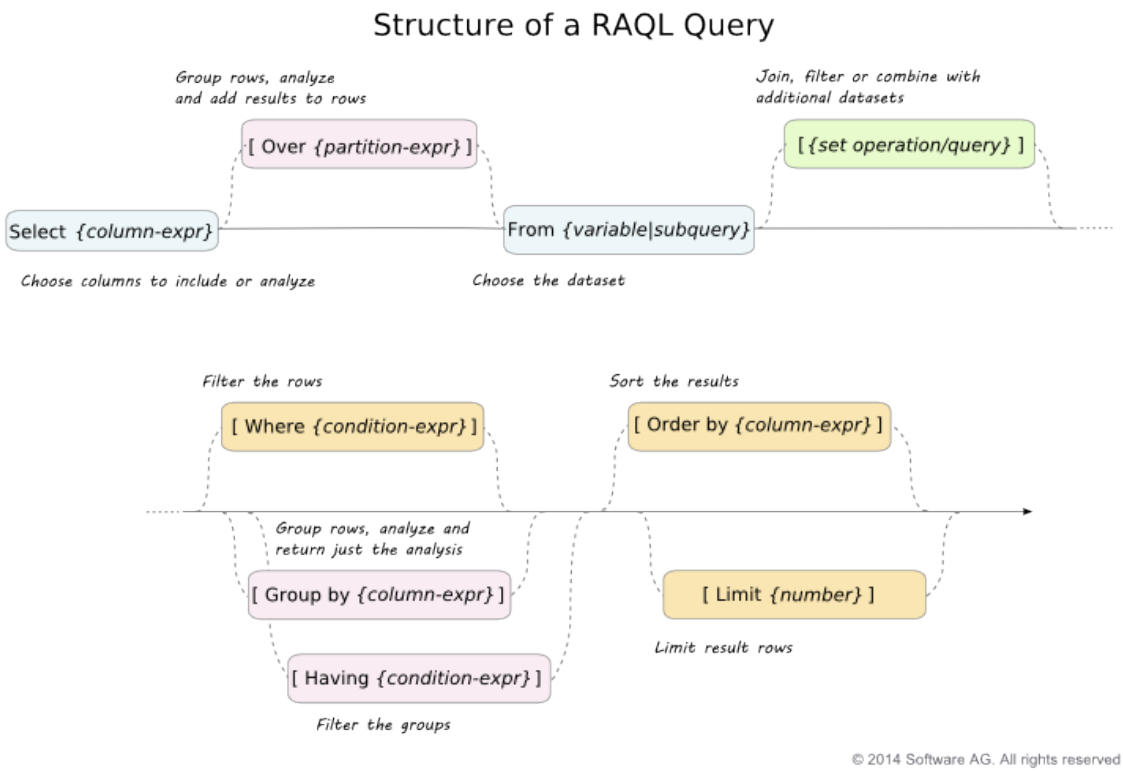
- *Databases*: using `<sql>`.
- *Snapshots of MashZone NextGenMashables or Mashups*: using `<snapshot>`.
- *Datasets Stored in one of the MashZone NextGen Analytics In-Memory Storesin BigMemory*: using `<loadfrom>`. Datasets may be stored by MashZone NextGen, using `<storeto>`, or by external systems using BigMemory APIs.

The RAQL Query Syntax

Once you have the dataset loaded, you can use RAQL queries to analyze and manipulate the data. RAQL is very similar to the Structured Query Language (SQL) used to access data in a database. It is:

- Case-insensitive. `WHERE`, `where` and `WhEre` are all valid.
- A focused subset of SQL features. Some SQL clauses and syntax are not found in RAQL.
- Designed to handle streaming data.
- Works only with flat, simple data, discussed previously in "Structure, Format and Access to Datasets with RAQL" on page 691.

Like SQL, a RAQL query is composed of clauses:



We will explore a simple use of most of these query clauses in this topic. See "RAQL Queries" on page 704 for a synopsis of the valid expressions for each of these clauses along with links to other examples.

As with SQL, the Select and From clauses are required. All other clauses are optional.

The Select clause determines which columns to include in the result and can also perform analysis when it is used with either the Over clause or the Group By clause. The From clause determines which dataset to query, or can define a subquery to use as the source of data.

Set operation clauses allow the query to retrieve an additional dataset, using another query, and then join, combine or filter these datasets to derive a more complex dataset. The Join set operation, for example, matches rows in both datasets based on a condition and adds columns from both datasets to the joined row.

The Where clause filters rows from the dataset. Order By sorts the result rows. And Limit determines the maximum number of result rows that the query can return.

The Over and Group By clauses both group dataset rows into different sets based on an expression. These groups determine the scope of rows that are used in analytic functions in the Select clause. The Having clause filters the set of groups that are returned in a Group By clause.

Over and Group By are mutually exclusive as they have different affects on the data returned by the query. The Over clause performs calculations and adds the calculations as additional columns to each row. Group By instead performs calculations and returns just the calculations for each group.

Most RAQL clauses also support the use of functions within their expressions. RAQL functions come in two varieties:

- *Plain* functions, that perform some simple transformation to the values of a column for each row, such as `upper()` to change text to upper-case.
- *Analytic* functions, more commonly known as aggregate or window functions, perform calculations using multiple rows in a group, partition or window defined in the Over or Group By clauses.

  Aggregate analytic functions use all rows in the current scope, such as `sum()`. While window analytic functions use specific rows, such as `rownumber()`. These functions include simple arithmetic as well as statistical functions, machine learning functions or other analysis algorithms.

RAQL provides a set of built-in functions (plain and analytical) as well as a way for you to define your own functions. See "Built-In RAQL Functions" on page 775 for more information.

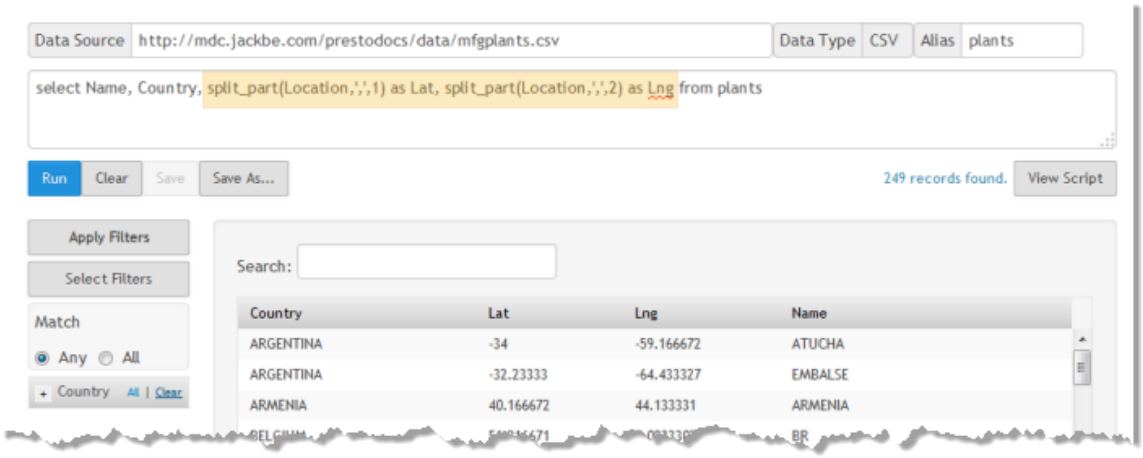
Use Plain Functions to Update, Select or Sort Rows

You can apply plain functions to individual columns in a dataset stream in any clause in a RAQL query. Plain functions can update data in each row in the Select clause, help to filter rows in Where conditions or help to sort results in Order By. They can also be

used along with analytic functions in Over or Group By clauses or in subqueries in From clauses.

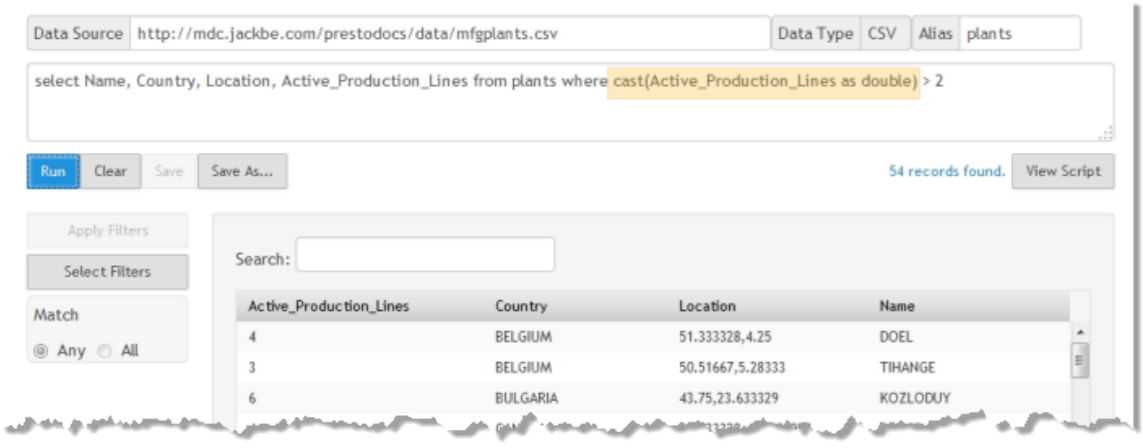
MashZone NextGen includes a set of built-in plain functions. See "Built-In RAQL Functions" on page 775 for details. You may also have additional plain, user-defined functions available.

The first example uses the built-in plain function `split_part` in the Select clause, to split longitude and latitude data into two columns for the Manufacturing Plants dataset.

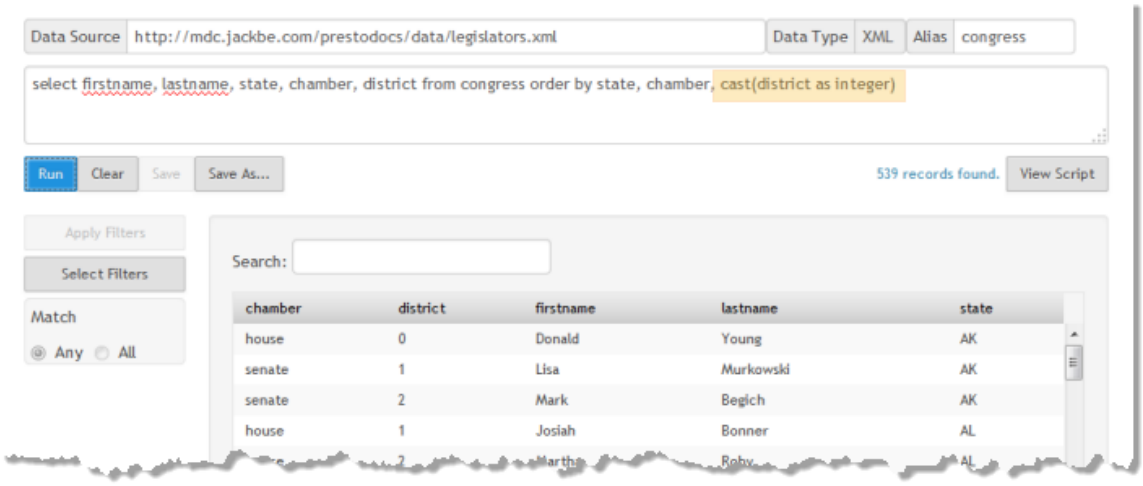


The Select clause uses the `split_part` function on the Location column to split the data into two separate pieces before and after a comma delimiter. The first call extracts the latitude and the second call extracts the longitude. In each case the result of the function is added to the query results as a separate column using `as` *alias* to provide the name of the new column.

The next example uses the `cast` function in a Where clause to ensure that the Active_Production_Lines column is treated as a number for filtering. Casting functions simply cast the data in the column to an appropriate datatype.



The last example uses the sample dataset for US legislators previously introduced in A Basic RAQL Query. It sorts legislators by state and district, using the `cast` function to ensure that their district is sorted numerically:



Datatype Information for Loaded Datasets

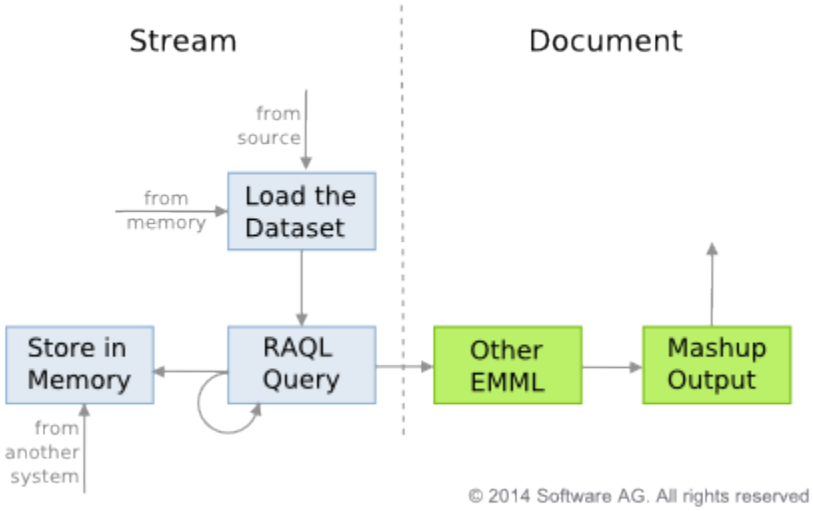
The plain functions used to cast the datatype of a column in the previous section point out one issue to be aware of for RAQL queries: in many cases, the RAQL Engine that runs the query has no datatype information for a given column. This is particularly true for data in CSV or XML formats.

When datatype information is not available, the RAQL Engine derives a schema based on the first row of data. For most purposes, this means that the data is treated as a string unless you explicitly cast it to another datatype.

Instead of using casting functions, you can provide datatype information for datasets in your mashup. See "Providing Dataset Path and Datatype Information in a Schema" on page 722 for more information.

The Stream/Document Boundary

The examples so far have used an EMML statement to load the data as a stream, used RAQL extension statements to query the stream and then returned the query results as the mashup results. Currently, however, mashups *cannot return streams* as their results. As the following figure shows, streams must be converted to a document before being returned from the mashup.



You must also convert query result streams to documents to use EMML statements that are not RAQL extensions with query results. Any EMML statement that uses XPath expressions requires that the stream first be converted to a document.

Note: Because of the volume of data, it is a good practice to *avoid* using the `<display>` statement with RAQL query results.

The `<raql>` statements used in earlier examples implicitly converted the query results to a document-type variable by using `result` as the output variable and not setting the streaming mode. You can have query results returned in a stream, when needed, by setting the `stream` attribute on the RAQL query. For example:

```
<!-- performs query and returns results as a stream -->
<raql outputvariable="$queryresult" stream="true" >
  select firstname, lastname, state, chamber, party, gender
  from congress where chamber='Senate'
</raql>
```

In general, if a mashup statement or the receiving variable does not set `stream='true'`, then the dataset will be treated as a document not a stream.

Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics

So far, the examples have actually not used a large dataset that would require significant memory for MashZone NextGen. The examples have all loaded data directly from original sources without using Terracotta BigMemory or the MashZone NextGen Analytics In-Memory Stores. But this is not realistic in many cases.

The previous figure (in The Stream/Document Boundary section) also illustrates the basic flow when you do need to store the dataset in memory to handle large amounts of data:

- A mashup can stream data from an original source, preprocesses this stream if needed, and then stores the stream in an In-Memory Store.
- Other systems may also store data directly in an In-Memory Store that has already been declared.
- Other mashups can then load data from memory and perform further queries or analysis as needed.

Mashups that store datasets in MashZone NextGen Analytics In-Memory Stores use the `<storeto>`EMML extension statement. The dataset to store *must also* be the results of a RAQL query, although the query can simply select the entire dataset.

Note: New in this release, other external systems can also store datasets in In-Memory Stores directly if the store is *declared*. See "Declared Versus Dynamic In-Memory Stores" on page 857 for more information.

Each dataset is streamed to one In-Memory Store with a unique name that other mashups can then connect to to load this dataset. This store may already exist or the `<storeto>` statement may create the In-Memory Store and then store the dataset. With existing stores, the dataset being stored may be appended to the existing data or it may replace all existing data.

The following example mashup retrieves performance data for stocks from a URL using `<directinvoke>`. It uses a RAQL query to package all the data for storage with `<raql>` and includes `stream='true'` to treat the query results *as a stream*.

```
<mashup name='storeStockDataset'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'>
  <output name='result' type='document' />
  <variable name='diResult' type='document'/>
  <variable name='stocksDS' type='document' />
  <directinvoke method='GET' stream='true' outputvariable='diResult'
    endpoint='http://mdc.jackbe.com/prestodocs/data/stocks.xml'
    timeout='5' onerror='abort' />
  <raql stream='true' outputvariable='stocksDS'>
    select * from diResult
  </raql>
<storeto cache='stocks2011' key='#unique' variable='stocksDS'      version='2.0'/>
</mashup>
```

The mashup finally stores the data selected in the query in an In-Memory Store named `stocks2011`. If this store does not already exist, MashZone NextGen creates this In-Memory Store dynamically when you run the mashup.

Note: In-Memory Stores can also be declared before they are used, providing more configuration and storage options. See "Declared Versus Dynamic In-Memory Stores" on page 857 for more information and "Store Data in MashZone NextGen Analytics In-Memory Stores" on page 865 for an example.

Each row of the dataset is assigned a unique key based on the method defined in the `key` attribute. If the store does already exist and contains data, this dataset is appended to any existing data by default.

This example did not filter or adjust the dataset in any way before storing it. But you can also use a RAQL query to preprocess the data you want to store in MashZone NextGen Analytics In-Memory Stores.

Once you, or an external system, store a dataset, other mashups can use the `<loadfrom>`EMML extension statement to load this dataset stream for queries and other processing. The following example shows a mashup to retrieve this stock dataset and return just the first 10 rows:

```
<mashup name='simpleStocksLoad'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'>
  <output name='result' type='document' />
  <variable name='stocks' type='document' stream='true' />
  <loadfrom cache='stocks2011' variable='stocks' version='2.0'/>
  <raql outputvariable='result'>
    select * from stocks limit 10
  </raql>
</mashup>
```

The stored dataset from these example, shown below, will be used in other examples in Getting Started to discuss Group By and Over query clauses for grouping and analysis:

Group and Analyze Rows

To analyze the data in a dataset stream you can use the Group By clause or the Over clause. Group By, as in SQL, categorizes rows into sets based on the unique values of one or more columns. The analysis then is performed on each group defined by the analytic function(s) that are used in the query's Select clause.

In this example, we group the stocks dataset that was stored in an In-Memory Store in the previous section and determine the highest price for each stock symbol in each year. This mashup uses `<loadfrom>` to retrieve the stock dataset stream from the in-memory store and then issues the RAQL query.

```
<mashup name="stocksBySymbolYear"
 xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
 xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
 xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation

  <output name="result" type="document"/>
  <variable name="stocks" stream="true" type="document"/>

  <loadfrom cache="stocks2011" variable="stocks" version="2.0"/>

  <raql outputvariable="result">
  select symbol, extract_year("date"), max(high) as highest
  from stocks
  group by symbol, extract_year("date")
  order by symbol, extract_year("date")
 </raql>
</mashup>
```

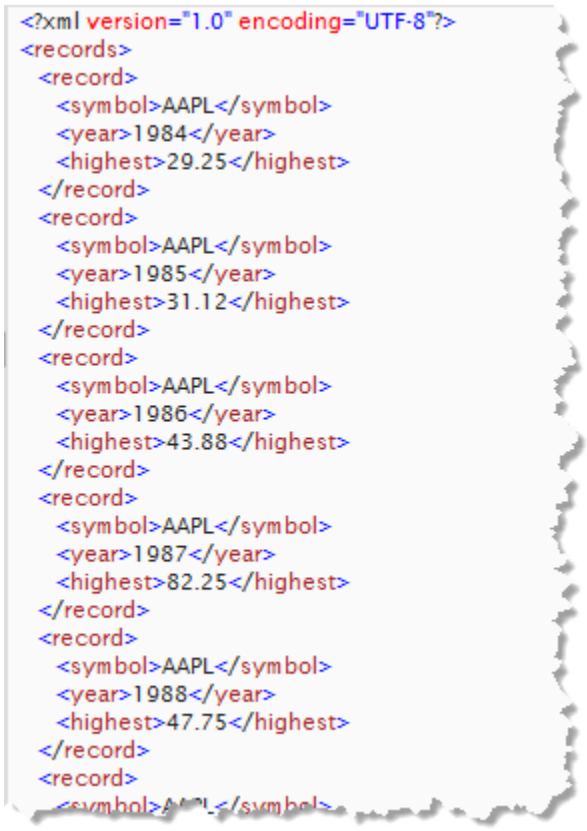
The Group By clause uses a list of column expressions to determine how rows are grouped. This can be as simple as one column, although it is quite common to group by two or more. Unique values from the combination of columns then determine which group a given row belongs to.

This query uses the plain function `extract_year()` to extract the year for each row from the date column and the `max()` analytic function in the Select clause to discover the highest price for all rows in each group. Because Group By returns a single row for each group, you must use aggregate analytic functions which perform calculations for all values in the current scope (group in this case) and return a single value.

Note: This is also an example of escaping column or other names that conflict with RAQL reserved words. In this case, the `date` column name is a conflict, so the name is enclosed in quote marks. See "RAQL Reserved Keywords" on page 729 for a list.

RAQL has a set of built-in analytic functions that you can use in Group By clauses, or you can write and add your own analytical functions. See "Built-In RAQL Functions" on page 775 for more information.

The results of this query using Group By, shown here, contain one row for each symbol + year combination:



```
<?xml version="1.0" encoding="UTF-8"?>
<records>
  <record>
    <symbol>AAPL</symbol>
    <year>1984</year>
    <highest>29.25</highest>
  </record>
  <record>
    <symbol>AAPL</symbol>
    <year>1985</year>
    <highest>31.12</highest>
  </record>
  <record>
    <symbol>AAPL</symbol>
    <year>1986</year>
    <highest>43.88</highest>
  </record>
  <record>
    <symbol>AAPL</symbol>
    <year>1987</year>
    <highest>82.25</highest>
  </record>
  <record>
    <symbol>AAPL</symbol>
    <year>1988</year>
    <highest>47.75</highest>
  </record>
  <record>
    <symbol>AAPL</symbol>
```

The complete EMML for this mashup is:

```
<mashup name='stocksBySymbolYear'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'>
  <output name='result' type='document' />
  <variable name='stocks' type='document' stream='true' />
  <loadfrom cache='stocks2011' variable='stocks'/>
  <raql outputvariable='result'>
    select symbol, extract_year("date"), max(high) as highest
    from stocks
    group by symbol, extract_year("date")
    order by symbol, extract_year("date")
  </raql>
</mashup>
```

Group and Analyze Rows with Row Detail

The other RAQL query clause that you can use to perform analysis is the Over clause. Like Group By, the Over clause segments the rows of the dataset into different groups, known as *partitions*. The primary differences between Group By and Over are:

- All rows of the dataset that meet the conditions of the Where clause, if any, are returned from a query with an Over clause, rather than just one row per group.
- You can also define *windows* within a partition. A window consists of the current row within a partition and the number of preceding and following rows you define.
- Analytic functions are applied to either the full partition or to each window within the partition. The results of the analysis is added as a new column to either each row in the partition or to the current row for each window.
- The results of analytic functions can also be running calculations, such as running totals, including the current row and all preceding rows.

Let's look at a simple partition example. The following mashup loads the stock dataset from an In-Memory Store (stored earlier in Getting Started).

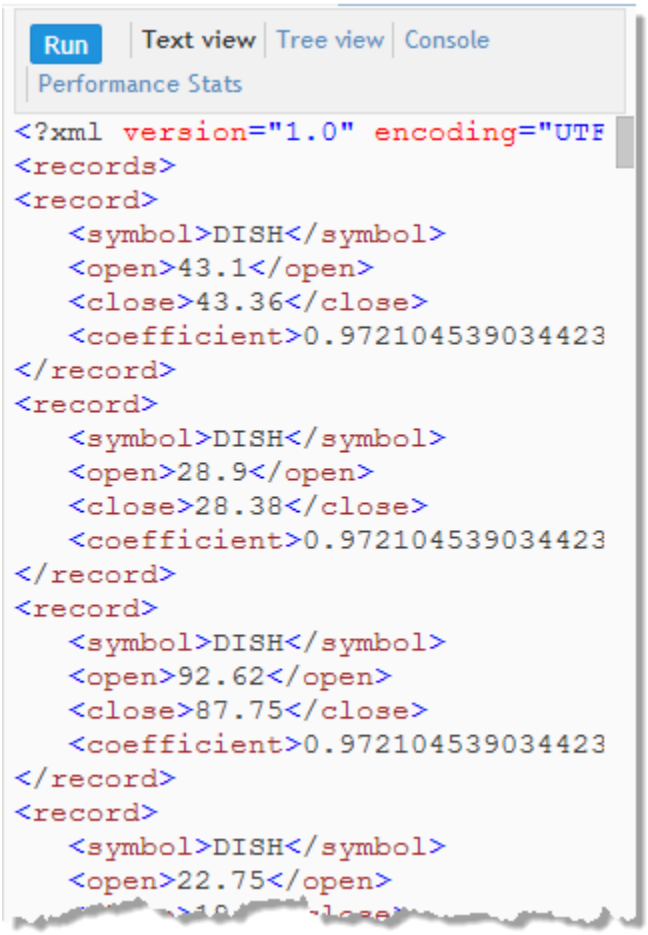
```
<mashup name='stockPriceCorrelation'
        xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
        xsi:schemaLocation='http://www.openmashup.org/schemas/v1.
        xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
        xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLM
        xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtens

  <output name='result' type='document' />
  <variable name='stocks' type='document' stream='true' />

  <loadfrom cache='stocks2011' variable='stocks'/>

  <raql stream='false' outputvariable='result'>
    select symbol, open, close,
    corr(open,close) over (partition by symbol) as coefficient
    from stocks
    where symbol like 'D%' or symbol like 'N%'
  </raql>
</mashup>
```

The Select clause selects each column that will be used in the calculation or in the definition of the partition (`symbol`, `open` and `close`).

Lastly, Select uses the built-in `corr` analytic function to determine if there is a linear correlation between opening and closing prices for each symbol. The Over clause defines the partitions that this analytic function is applied to.

The Where clause filters the rows that are included in each partition to specific stock symbols based on a name pattern. With this dataset, this limits the results to the symbols `DISH` and `NFLX`.

The results of this query, shown below, include each row for the selected symbols and a new column, `coefficient`, that includes the result of this analysis function:



RAQL has a set of built-in analytic functions that you can use in Over clauses, or you can write and add your own analytic functions. See "Built-In RAQL Functions" on page 775 for more information.

The complete EMML for this mashup is:

```
<mashup name='stockPriceCorrelation'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'>
  <output name='result' type='document' />
  <variable name='stocks' type='document' stream='true' />
  <loadfrom cache='stocks2011' variable='stocks'/>
  <raql outputvariable='result'>
    select symbol, open, close,
    corr(open,close) over (partition by symbol) as coefficient
    from stocks
    where symbol like 'D%' or symbol like 'N%'
  </raql>
</mashup>
```

Where to Go Next

This finishes the basic examples of RAQL queries and how to work with large datasets using MashZone NextGen Analytics. For examples of other query techniques or advanced capabilities, such as dynamic queries, see "RAQL Queries" on page 704 and "Working with MashZone NextGen Analytics In-Memory Stores" on page 857. For more information on the extensions that RAQL adds to EMML, see "RAQL Extension to EMML Statements for Mashups" on page 876.

RAQL Queries

The basics of using RAQL to query and analyze large datasets is discussed in "Getting Started with MashZone NextGen Analytics" on page 689. For specific techniques on loading large datasets or using specific RAQL query clauses, see the links in these sections:

- Techniques to Load Data
- Select Techniques
- From Techniques
- Over Techniques
- Where Techniques
- Group By Techniques
- Having Techniques
- Order By Techniques
- Multi-DataSet Operations

See also "Escape Characters for RAQL Queries" on page 710, "RAQL Operators" on page 773, "RAQL Datatypes and Data Formats" on page 771, "Built-In RAQL Functions" on page 775 and "Create and Add User-Defined Functions for RAQL Queries" on page 804.

Techniques to Load Data

To load datasets from:

- Files, see "A Basic RAQL Query" on page 689.
- <directinvoke>, see .
- <sql>, see "Load Data with <sql>" on page 710.
- <invoke>, see "Load Data with <invoke>" on page 712.
- Snapshots, see "Load Snapshot Data with <snapshot> or <raql>" on page 714.

- The in-memory store, see "Load Data from the MashZone NextGen Analytics In-Memory Stores" on page 872 and "Store Data in MashZone NextGen Analytics In-Memory Stores" on page 865.

Select Techniques

	Column Expression	Example
`select`	*`column-name[, column-name ,...]`*	To select specific columns, see "A Basic RAQL Query" on page 689
	`*`	To select all columns, see
	`function(column-name)`	To use plain functions to update selected column data, see "Use Plain Functions to Update, Select or Sort Rows" on page 693 and "Plain Functions on Select Columns" on page 720.
	`column-name``distinct`	To select distinct values from a column, see "Select Distinct Values" on page 718.
	`column-name``as` *`new-name`*	To provide names for calculations or columns, see "Alias Names for Columns or Calculations" on page 719.

Over Techniques

	Column Expression	Example
`over`	`()`	To use the entire dataset as one partition, see the middle subquery in "From Subqueries" on page 731.
	`(partition` *`column-name`*`)`	To group with a simple partition, see "Group and Analyze Rows with Row Detail" on page 701

	Column Expression	Example
	`(partition column-name`between rows `n`preceding and `n`following)	To apply a window to each row within a partition, see "Partitions and Windows" on page 754
	`(partition column-name`order by `column-name`)	To obtain running calculations in partitions, see "Running Aggregates" on page 757.
		To provide an index of each row or ranking within each partition, see "Number or Rank Rows" on page 758.
	any-partition-expression	For examples of statistical analysis, see "Analytic Functions for Partitions and Windows" on page 759.

From Techniques

	Column Expression	Example
`from`	*variable-name*	See "A Basic RAQL Query" on page 689.
	variable-name/path-to-row	See "Load Data with <invoke>" on page 712 and also "Dataset Paths, Names and Datatypes" on page 721.
	subquery	See "From Subqueries" on page 731.
	variable-expression`limit n`	To limit the total number of rows returned, see "Limit the Rows Returned" on page 732.

Where Techniques

Clause	Condition Expression	Example
`where`	*`simple-condition`*	For a simple filter condition, see .
	`complex-condition`	To use multiple conditions, arithmetic or logical operators, see "Where Complex Conditions" on page 733.
	`column-name``like` *`text-pattern`*	To use text matching, see "Where Text Like Patterns" on page 736.
	`column-name``in (`*`value1, value2[,...]`*`)`	To match to an enumerated set of values, see "Where in Sets" on page 737.
	`column-name``between` *`range-start`*`and` *`range-end`*	To match within a range of values.
	`dynamic condition`	To provide dynamic values to filter conditions, see "Parameters in Where Clauses" on page 739.
	`function(column-name) in condition`	To use functions on column values in filters, see "Plain Functions in Where" on page 740.

Group By Techniques

Clause	Column Expression	Example
`group by`	*`column-name`*	For simple grouping, see "Group and Analyze Rows" on page 699.
	`complex-expression`	For complex grouping, see "Multi-Level Group Calculations" on page 743.

Having Techniques

Clause	Column Expression	Example
`having`	*`aggregate-condition`*	A single or complex condition based on the aggregate calculations for each group.

Order By Techniques

Clause	Column Expression	Example
`order by`	*`column-name`*	For simple sorting, see "Use Plain Functions to Update, Select or Sort Rows" on page 693.
	`column-name, column-name`	For multiple-level sorting, see "Sort Directions and Multi-Level Sorts" on page 741.
	`function(column-name)`	To use functions on column values in sorts, see "Plain Functions in Sorts" on page 742.

Multi-DataSet Operations

Clause	Column Expression	Example
`[inner] join`	*`query`*`on condition`	To join the columns of each row from two or more queries based on the specified condition. This is an inner join where joined results contain only those rows that have matches from all the queries involved in the join.
	`query``using columns-list`	To join the columns of each row from two or more queries based on matches for the column(s) specified. Each query result must

Clause	Column Expression	Example
		contain columns that match this list.
`natural join`	*query*	To join the columns of each row from two or more queries based on equivalent values in the columns in query results with matching names. Columns with matching names must contain data of the same type.
`cross join`	*query*	Creates a Cartesian product matching each row from one query with every row from one or more additional queries and including all the columns.
`union`	`[distinct]` *query*	Merges the results of two of more queries including every row. Each query must contain the same columns with the same datatype. The `distinct` keyword is optional as duplicate rows are removed from the result by default.
	`all` *query*	Merges the results of two of more queries including every row (duplicates are not removed). Each query must contain the same columns with the same datatype.
`intersect`	`[distinct]` *query*	Includes each row from the results of the first query that has a matching row in the results of the second query. The `distinct` keyword is optional as the default behavior is to remove duplicate rows from the first query results.
	`all` *query*	Includes all rows, including duplicates, from the results of the first query that have a matching row in the results for the second query.

Clause	Column Expression	Example
except minus	[distinct] *query*	Includes each row from the first query that does *not* match any row in the second query. The distinct keyword is optional as the default behavior is to remove duplicate rows from the first query results.
except minus	all *query*	Includes all rows, including duplicates, that do *not* match any row in the second query.

Escape Characters for RAQL Queries

You can enter RAQL queries as either the value of the the <raql> element in a mashup or as the value of the query attribute in <raql>. Because the query is in an XML file (the mashup), you must use the escaped form of the following XML delimiters when they appear in a RAQL query:

- & use &
- < use <
- > use >

In addition, you may have to escape either the single or double quote mark when they appear in the query if that quote mark is used as the delimiters for the query attribute. The escaped forms are:

- " use "
- ' use '

For example:

```
<raql query="select name,descr,&quot;date&quot; from myVar" outputvariable="result"/>
```

Or:

```
<raql query='select name,descr from myVar where descr like &apos;Arnold%&apos;' outputvariable="r
```

Load Data with <sql>

You can also load a dataset directly from a database using the <sql> statement in EMML and then use RAQL to perform analysis. To query a dataset from a database you must:

1. Add a Datasource to MashZone NextGen to Connect to the Database
2. Stream the Database Results and Query with RAQL

Add a Datasource to MashZone NextGen to Connect to the Database

If you do not already have a datasource defined in MashZone NextGen for the database containing the data you want to work with, have your MashZone NextGen administrator add one.

The example shown in this topic is from a MySQL database with machine sensor data. Every half second various readings are added to this database for different devices. The dataset, shown below in XML format, includes the value (data_value) for a reading, a code (data_item_id) for the type of reading, a code for the device (device_id) and both a timestamp and a date/time in milliseconds when the reading was taken.

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
  <record>
    <device_id>876</device_id>
    <data_item_id>80</data_item_id>
    <data_time>1333291861000</data_time>
    <data_value>8.5</data_value>
    <t_stamp>2012-04-01 14:51:01.0</t_stamp>
    <milli_sec>0</milli_sec>
  </record>
  <record>
    <device_id>876</device_id>
    <data_item_id>80</data_item_id>
    <data_time>1333291891000</data_time>
    <data_value>9.6</data_value>
    <t_stamp>2012-04-01 14:51:31.0</t_stamp>
    <milli_sec>0</milli_sec>
  </record>
  <record>
    <device_id>876</device_id>
    <data_item_id>80</data_item_id>
    <data_time>1333291921000</data_time>
    <data_value>9.7</data_value>
    <t_stamp>2012-04-01 14:52:01.0</t_stamp>
    <milli_sec>0</milli_sec>
  </record>
  <record>
```

Stream the Database Results and Query with RAQL

Once you have an available datasource to connect to the database, use the `<sql>` statement in a mashup to query the database for the dataset and *set streaming on* with the `stream` attribute. The name of the datasource is set in the name attribute.

The following example queries for sensor data for a single type of reading across all devices. This also sets a `fetchsize` that is used to stream the dataset to MashZone NextGen.

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/..
      xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
      xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
      xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
      name='sensorID80'>

 <output name='result' type='document' />
 <variable name="sensors" type="document" stream="true"/>

 <sql name='Sensor DB' fetchSize='1000' outputvariable='sensors' stream='true'
    query='select * from sensors.readings_raw where data_item_id=80'/>

 <raql stream='false' outputvariable='result'>
  select device_id, data_item_id, data_value, t_stamp
  from sensors
  where t_stamp > "date"("2012-04-01 23:59:59", "yyyy-MM-dd HH:mm:ss") and
        t_stamp &lt; "date"("2012-04-03 00:00:01", "yyyy-MM-dd HH:mm:ss")
 </raql>
</mashup>
```

Use `<raql>` to further query and analyze the dataset just as you would with XML or CSV datasets. The results of the database query are already in a flat, table structure that matches the RAQL data model, so no additional path information is needed.

Query and Store to the MashZone NextGen Analytics In-Memory Stores

You can also query databases and then load the dataset directly into the in-memory store.

Load Data with <invoke>

You can load datasets, for analysis with RAQL, from any MashZone NextGen mashable or mashup using `<invoke>`. Mashable or mashup results must be a document.

Note: Currently, `<invoke>` does not support dataset streaming, so large datasets may present performance issues.

The following example loads results from the Yahoo local search mashable, one of the sample mashable you may register when you install MashZone NextGen:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas
/EMMLPrestoSpec.xsd'
xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
name='invokeRAQLQuery'>

  <output name='result' type='document' />
  <variable name="appid" type="string"
    default=".kcC72DV34FYTpAGuwwbV8YGI.DsMBQ0RB9eZARS621ecnHq33c.g1XJV93a64hrdaM3" />
  <variable name="query" type="string" default="coffee" />
  <variable name="zip" type="string" default="98102" />
  <variable name="results" type="string" default="20" />
  <variable name="searchResults" type="document"/>

  <invoke service="YahooLocalSearchREST" operation="getData" outputvariable="searchResults"
  inputvariables="appid,query,zip,results" />

  <raql outputvariable='result'>
    select Title, Address, City, State, Distance from searchResults/ResultSet/Result
  </raql>
</mashup>
```

Some points to note with this example:

- Streaming is *not* set for either the `searchResults` variable that holds the results of invoking Yahoo local search or for the `<invoke>` statement itself.
- RAQL executes the query even though the result dataset is not streamed.
- To help clarify the exact XML elements in the mashable results that are considered a row for the RAQL query, the From clause uses both the variable name for the dataset and the full path to the element that is a row. Paths are sometimes useful to clarify rows when datasets are XML. See "Dataset Paths, Names and Datatypes" on page 721 for more information.

The results for Yahoo local search and the results from the subsequent query are shown below:

Yahoo local search results

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet xmlns="urn:yahoo:lcl" xmlns:xsi="http://www
        firstResultPosition="1"
        totalResultsAvailable="679"
        totalResultsReturned="20"
        xsi:schemaLocation="urn:yahoo:lcl http://local.y
 <ResultSetMapUrl>http://maps.yahoo.com/broadband
 <Result>
    <Title>Whidbey Coffee Company</Title>
    <Address>411 15th Ave E</Address>
    <City>Seattle</City>
    <State>WA</State>
    <Phone>(206) 325-6520</Phone>
    <Latitude>47.622289</Latitude>
    <Longitude>-122.312764</Longitude>
    <Rating>
      <AverageRating>5</AverageRating>
      <TotalRatings>2</TotalRatings>
      <TotalReviews>2</TotalReviews>
      <LastReviewDate>1141158041</LastReviewDate
    </Rating>
    <Distance>1.08</Distance>
    <Url>http://local.yahoo.com/info-22149417-whidbe
    <ClickUrl>http://local.yahoo.com/info-22149417-w
    <MapUrl>http://local.yahoo.com/info-22149417-wh
```

RAQL query results

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
  <record>
    <Title>Whidbey Coffee Company</Title>
    <Address>411 15th Ave E</Address>
    <City>Seattle</City>
    <State>WA</State>
    <Distance>1.08</Distance>
  </record>
  <record>
    <Title>Starbucks</Title>
    <Address>1600 E Olive Way</Address>
    <City>Seattle</City>
    <State>WA</State>
    <Distance>1.11</Distance>
  </record>
  <record>
    <Title>Voxx Coffee</Title>
    <Address>2245 Eastlake Ave E</Address>
    <City>Seattle</City>
    <State>WA</State>
    <Distance>0.29</Distance>
  </record>
  <record>
    <Title>Analog Coffee</Title>
    <Address>235 Summit Ave E</Address>
```

All of the direct children of `<Result>` in the Yahoo local search results are accessible as columns in the RAQL query. Currently however, the following types of XML content are not accessible:

- Ancestors of rows, such as `<ResultSetMapUrl>`
- Descendants of row columns, such as `<AverageRating>`
- Attributes, such as `ResultSet/@TotalResultsAvailable`

Load Snapshot Data with <snapshot> or <raql>

You can load snapshots of mashable or mashup results as dataset streams for analysis in RAQL queries using the:

- `<snapshot>`EMML extension statement. This statement both creates a streaming variable to hold the snapshot dataset and executes the SQL query to retrieve snapshots from the MashZone NextGen Snapshot Repository. See "Load Snapshots in a Named Variable" on page 715 for an example.
- `<raql>`EMML extension statement with the `snapshotquery` attribute to load the dataset. This uses a local variable, named `snapshots` to hold the dataset stream which is only in scope for this RAQL query. See "Load Snapshots Anonymously" on page 716 for an example.

Before you can query snapshots, you must register the mashable or create the mashup with results you want to use. You must also run the mashable or mashup and take at least one snapshot.

Snapshot Queries

Queries to retrieve snapshots are SQL queries that *require* a Select, From and Where clause to define which snapshots to include in the dataset. You identify snapshots in the Where clause by one of these conditions:

- `service = '`*mashable ID or mashup ID*`'` for snapshots of results for mashables or mashups with one operation. See "Load Snapshots in a Named Variable" on page 715 for an example.
- `service = '`*mashable ID or mashup ID*`' and operation = '`*operation ID*`'` for snapshots of results for mashables that have multiple operations.
- `createdtime =` *datetime, date or time* for snapshots based on a time period. See "Choose Snapshots by a Time Period" on page 717 for an example.
- `job = '`*job ID*`'` for snapshots taken by a specific scheduled snapshot job. Job IDs include the mashable or mashup ID and the operation for that scheduled job plus the timestamp when the job was created. See "Choose Snapshots for a Scheduled Job" on page 717 for an example.

You can, of course, use different operators to specify the conditions and functions as needed.

Load Snapshots in a Named Variable

This query creates a variable, named `coffee`, to stream the selected snapshots and then use in further RAQL queries. The snapshot query retrieves all snapshots for a single mashable, the sample Yahoo local search mashable identified by ID:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
  name='coffeeSnaps'>

  <output name='result' type='document' />
  <snapshot name="coffee" query="select * from snapshots where service='YahooLocalSearchREST' "/>
    <raql outputvariable="result">
      select Title,Address,City,State,Latitude,Longitude,Distance from coffee limit 60
    </raql>
</mashup>
```

Once the dataset has been loaded, the RAQL query can act on the dataset or it can be stored in the MashZone NextGen Analytics In-Memory Stores.

The full EMML for this sample is:

```
<mashup name="coffeeSnaps"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openmashup
  <operation name="Invoke">
    <output name="result" type="document"/>
    <snapshot name="coffee" query="select * from snapshots where service='YahooLocalSearchREST' "
    <raql outputvariable="result">
      select Title,Address,City,State,Latitude,Longitude,Distance
      from coffee limit 60
    </raql>
  </operation>
</mashup>
```

Load Snapshots Anonymously

You can also load snapshots as a dataset into a local, anonymous variable with the `snapshotquery` attribute in `<raql>`. This example is identical to the named example shown previously except that the query to load snapshots in a dataset stream is specified on `<raql>`.

```
1  <mashup name="coffeeAnonymousSnaps"
2    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
3    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
4    xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
5    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/../schemas
   /EMMLPrestoSpec.xsd">
6
7    <operation name="Invoke">
8
9      <output name="result" type="document"/>
10     <raql snapshotquery="select * from snapshots where service = 'YahooLocalSearchREST'"
11     outputvariable="result">
         select Title,Address,City,State,Latitude,Longitude,Distance from snapshots limit 60
12     </raql>
13   </operation>
14 </mashup>
15
16
```

The full EMML for this sample is:

```
<mashup name="coffeeAnonymousSnaps"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd">
  <operation name="Invoke">
    <output name="result" type="document"/>
    <raql outputvariable="result"
snapshotquery="select * from snapshots where service = 'YahooLocalSearchREST'">
      select Title,Address,City,State,Latitude,Longitude,Distance
      from snapshots limit 60
```

```
    </raql>
  </operation>
</mashup>
```

Choose Snapshots by a Time Period

This example shows a snapshot query to select snapshots prior to a specific date:

```
<mashup name="snapshotByDate"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/../schemas
/EMMLPrestoSpec.xsd">
  <operation name="Invoke">
    <output name="result" type="document"/>
    <snapshot name="day" query="select * from snapshots where DATE(createdtime)
&lt; DATE('2013-03-14') "/>
    <raql outputvariable="result">select * from day</raql>
  </operation>
</mashup>
```

Note: Because snapshot queries are SQL queries, the exact syntax and functions that are available for the query depend on the type of database in which the MashZone NextGen Snapshot Repository is hosted.

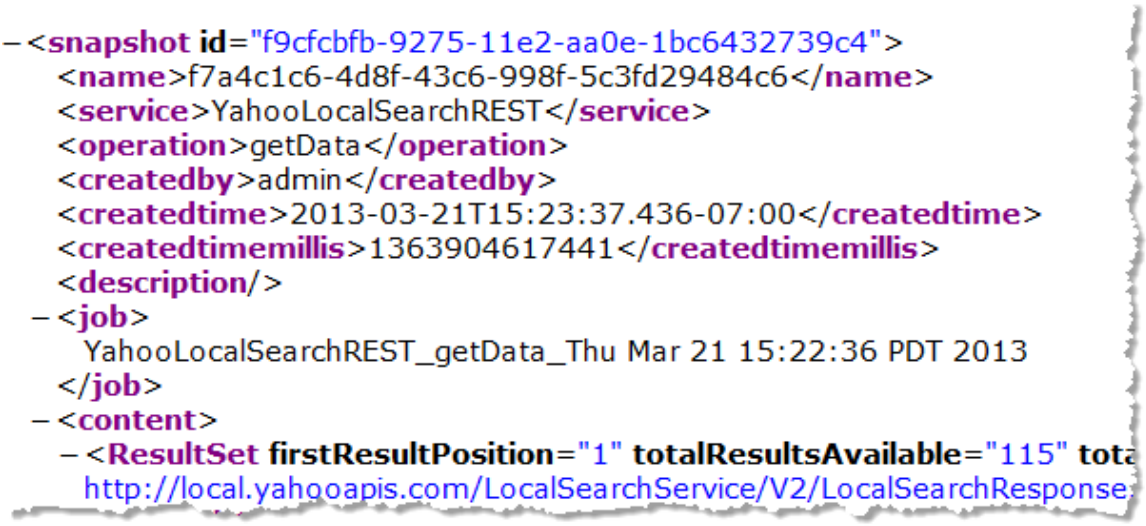
Choose Snapshots for a Scheduled Job

This example shows a snapshot query based on a schedule snapshot job:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas
/EMMLPrestoSpec.xsd'
xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
name='snapshotYahooJobs'>

  <output name='result' type='document' />

  <snapshot name='jobs' query="select * from snapshots where job =
'YahooLocalSearchREST_getData_Thu Mar 21 15:22:36 PDT 2013'" />

  <raql outputvariable='result'>
    select * from jobs
  </raql>
</mashup>
```

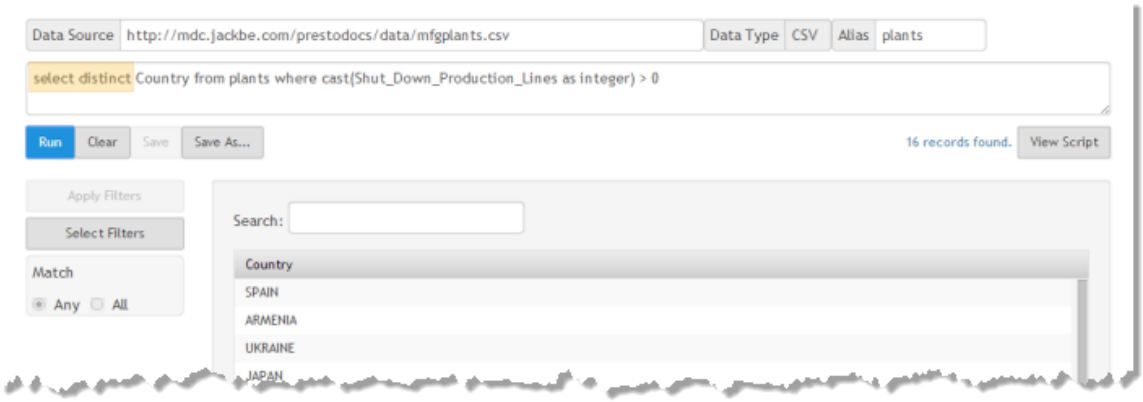
To find a snapshot job ID

1. Open the mashable or mashup artifact page and run the operation for that snapshot job.
2. Click the **Snapshots** tab and search to find one snapshot for that job.
3. Open the snapshot and click **Get XML** to see the raw snapshot data in a new browser window or tab:



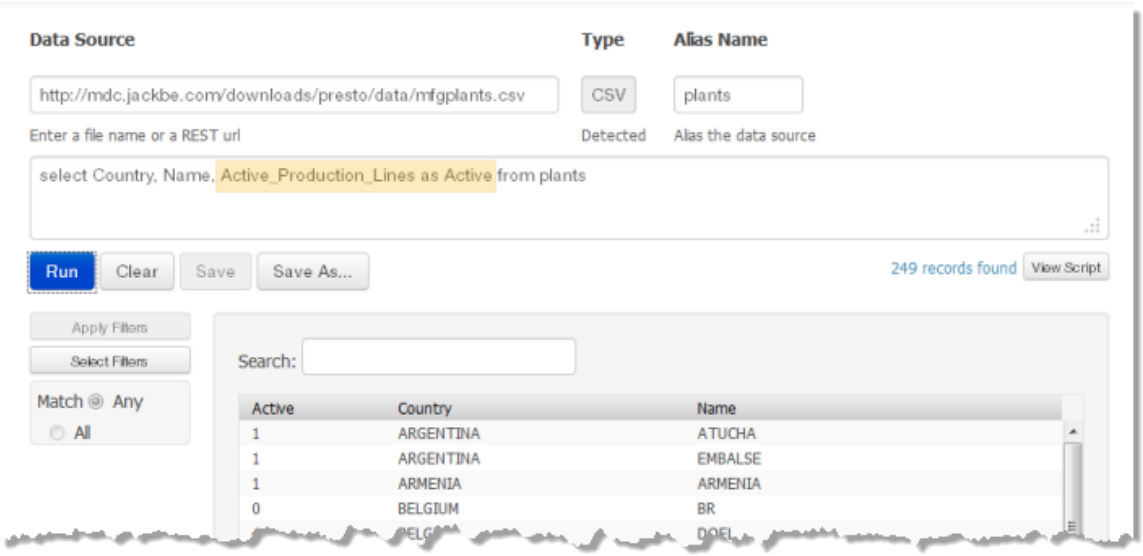
Select Distinct Values

Add the `distinct` keyword in Select clauses to retrieve only one row for each distinct value of a column. For example:



Alias Names for Columns or Calculations

You can change column names or supply column names for calculations using `as` in the Select clause, such as this example that changes a column name:



Examples of queries using `as` to provide column names for calculations are shown in Use Plain Functions to Update, Select or Sort Rows, Group and Analyze Rows and Group and Analyze Rows with Row Detail.

For more information, see also "Use Alias Names in Other Clauses" on page 719, "Use Alias Names to Handle HTML Column Content" on page 720 and "Valid Names for Datasets, Columns, Aliases, Paths and Functions" on page 727.

Use Alias Names in Other Clauses

The alias name for a column is defined in the Select clause. You *cannot* use this alias to refer to that column in any RAQL clauses, including expressions for later columns within Select, with the exception of the Order By clause. These RAQL queries, for example, will result in an error:

```
select Id, Descr, decimal(Qty) as Quantity from items where Quantity > 5
```

```
select Id, Descr, decimal(Qty) as Quantity, (Quantity / 1000) as
PerMille from items
```

However, this query is valid:

```
select Id, Descr, decimal(Qty) as Quantity from items order by Id,
Quantity
```

There are two techniques you can use rather than using an alias. You can duplicate the original column, such as this example:

```
select Id, Descr, decimal(Qty) as Quantity from items where decimal(Qty)
> 5
```

This can be cumbersome or in some cases impossible where the syntax is complex. Another solution is to define the alias in a subquery so that you can refer to it in an outer query. For an example of this syntax, see "From Subqueries" on page 731.

Use Alias Names to Handle HTML Column Content

Columns in XML datasets can sometimes contain HTML markup, such as <img> tags. This is quite common in data from RSS or Atom web feeds. These HTML tags in column data can cause information to be missing from RAQL queries or cause other problems because the tags are incorrectly interpreted.

To overcome these errors, you can have RAQL escape the column content with HTML tags using *CDATA sections* by using a column alias in the following form:

original-column-name `as` *original-column-name* `__cdata`

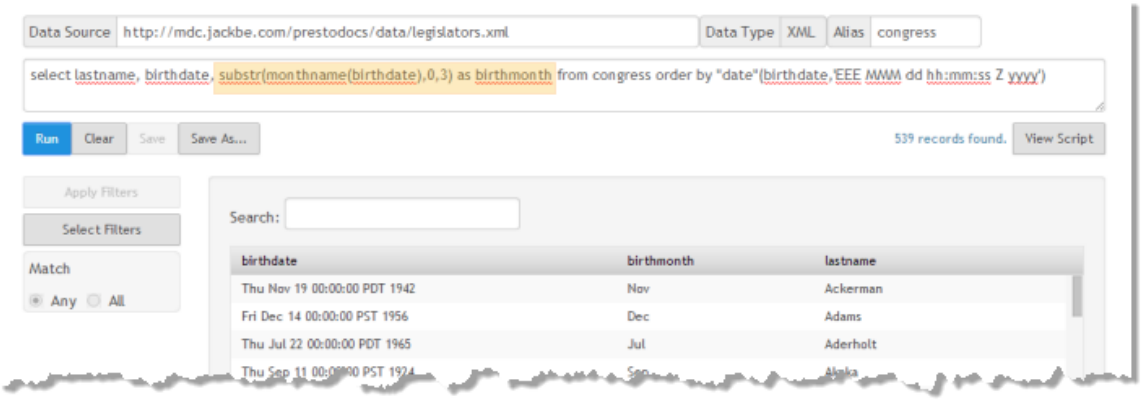
Note: This uses two underscores (_) in the alias name.

With a column named `title` for example, the alias name to escape any HTML markup in the column content would be `title__cdata`.

Plain Functions on Select Columns

Plain functions can be used in any RAQL query clause. You may use any MashZone NextGen built-in plain function (see "Built-In RAQL Functions" on page 775 for a list) or user-defined plain functions that you or other MashZone NextGen developers have added.

Simple examples of using plain functions in Select clauses are shown in Use Plain Functions to Update, Select or Sort Rows. Plain functions can also be nested, such as this example which uses both the `monthname` and the `substr` plain functions to derive a three-character English name for the birth month of each legislator:



Dataset Paths, Names and Datatypes

When working with XML or CSV datasets, there are three potentially troublesome areas that you can improve with specific techniques:

- The data model for XML datasets is frequently hierarchical, including additional metadata beyond the flat rows of interest to RAQL and adding additional layers of structure.

  To simplify queries, RAQL automatically attempts to detect which elements in an XML dataset should be considered rows. This allows you to refer to rows in the dataset in RAQL queries using only the name of the variable containing the dataset, such as:

  ```
  select firstname, lastname, state from congress
  ```

  The rules that govern automatic schema detection are shown in " Default rules for schema detection from XML" on page 721.

  In some cases, this default may not be the dataset elements you actually need to work with or query results may be incomplete. You can override this default by "Adding Paths to Clarify RAQL Row Detection" on page 722 or "Providing Dataset Path and Datatype Information in a Schema" on page 722.

- In some cases, you may also need to alter column names to make them valid for RAQL or for EMML. In cases with queries using multiple datasets, you may also need to clarify the specific context for column names.

  See "Valid Names for Datasets, Columns, Aliases, Paths and Functions" on page 727 for information.

- With XML or CSV data, RAQL has no metadata about the datatypes for each column so the data is treated as a string. You can fill this gap, to simplify the need for casting functions, by "Providing Dataset Path and Datatype Information in a Schema" on page 722.

Default rules for schema detection from XML

By inspecting the beginning of an XML document, schema detection automatically determines path expressions for

- repeating elements, locating row elements relative to the document root,
- column elements or attributes, locating columns relative to row elements.

This process begins by finding potential columns. Attributes and elements without children elements are considered potential columns.

Their parent elements are candidates for the repeating element. From those, the first path that occurs at least twice will be chosen. If there is none that occurs twice, the first one with the highest number of columns is chosen.

The relative paths of all children elements of the repeating element make up the column element path expressions.

Adding Paths to Clarify RAQL Row Detection

If you need to override the default XML elements in a dataset that RAQL treats as rows, you can add a specific *path* to the elements you need to the variable in the From clause. For this example:

```
select firstname, lastname, state
  from congress/response/legislators/legislator
```

The variable is `congress` and `/response/legislators/legislator` is the path to the element for rows.

The path uses XPath syntax, starting with a slash (/) and separating each level of elements within the hierarchy with a slash. See "Load Data with <invoke>" on page 712 for an example of a From clause using paths.

See also "Providing Dataset Path and Datatype Information in a Schema" on page 722 for an alternative way to set path information.

Providing Dataset Path and Datatype Information in a Schema

Schemas provide four types of metadata to simplify or improve how RAQL queries interacts with XML or CSV datasets:

- Path information to define what elements are rows for XML datasets.
- Path information that indicates where to find the column values in XML datasets.
- Datatype information for each column in the dataset for XML or CSV datasets. For columns that are a date type, this can also include the lexical format for the data. Text columns may be specified as being compared case-insensitively.
- The delimiter character and whether the dataset has column names in the first row for CSV datasets.

See "Dataset Schema Syntax" on page 722 for information on how to define a dataset schema.

The scope of this schema is based on where you define it: within a single mashup or as a global MashZone NextGen attribute that can be used in any number of mashups. See "Example 109. Dataset Schemas Defined in Mashups" on page 724 and "Global Dataset Schemas as MashZone NextGen Attributes" on page 725 for information.

Dataset Schema Syntax

Schemas for datasets define a variable name for the schema, a set of columns with datatype and optional format and path information and an optional set of options for a dataset. The schema syntax is in the form:

```
define dataset variable-name(
  [column-name datatype [path=column-path] [format|case_insensitive] [,
   column-name datatype [path=column-path] [format|case_insensitive] ]...
```

```
  ]
)
[with options option-name = value [,
              option-name = value ]...
]
```

For example:

```
define dataset stocks (
       symbol string case_insensitive,
       date datetime "yyyy-MM-dd",
       open decimal,
       close decimal,
       high decimal,
       low decimal,
       volume decimal path="volume[1]")
       with options record="/stocks/stock"
```

See "Valid RAQL Datatypes" on page 772 for the types you can use in dataset schemas.

The `format` metadata for date or time type columns accepts any lexical pattern that is valid for the Java `SimpleDate` class. For the most common patterns you can use, see the "Date Formatter" on page 362 function for the Transformer block in Wires.

When specifying `case_insensitive` , RAQL will perform any comparisons of column values case-insensitively.

The optional column path specifies an XPath expression that is evaluated relative to the row element for extracting the column value. If no path is specified, it defaults to the first child element matching the column name (as in the example above). It is only necessary to specify the column path if it diverges from this default, e. g. for selecting attribute values.

There are three different options you can specify:

- `record=`*/path/to/row* identifies the elements within the dataset, starting from the root, that should be used as rows. This uses the same syntax as paths you specify in a From clause, *excluding* the variable name.

  See "Adding Paths to Clarify RAQL Row Detection" on page 722 for more information.

- `delimiter="`*character* `"` identifies the delimiter used in CSV datasets when it is not the default delimiter (commas).

- `header=[true|false]` indicates whether CSV datasets have column names as the first row. The default is true.

The options are comma separated.

Examples:

```
define dataset
               ...with options header=false
define dataset
               ...with options header=false, delimiter=";"
```

Dataset Schemas Defined in Mashups

You can declare a dataset schema in the mashup that loads a dataset using the EMML<variable> statement and a type of schema. For example:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/
  ../schemas/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
  name='xmlSchema'>
  <output name='result' type='document' />
  <variable name="stockType" type="schema">
    define dataset stocks (symbol string,
    date datetime,
    open decimal,
    close decimal,
    high decimal,
    low decimal,
    volume decimal,
    adjclose decimal)
  </variable>
  <variable name="stocks" type="variable:stockType" stream="true"/>
  <directinvoke  method='GET' stream='true' outputvariable='stocks'
    endpoint='http://mdc.jackbe.com/prestodocs/data/stocks.xml'/>
  <raql outputvariable='result'>
    select symbol, "date", open, close, volume from stocks
    where extract_year("date") = 2011 order by close
 </raql>
</mashup>
```

The variable named stockType defines a schema for the stock dataset introduced in "Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics" on page 696 in Getting Started. This variable is then *referenced* in the variable named stocks, using a type of variable:stockType, that will hold the dataset once it is loaded. The type identifies the named variable containing the schema for this dataset.

The primary advantage of having the dataset defined is that RAQL queries now know datatypes so that filter conditions in Where, sorting criteria in Order By and functions or calculations in Over or Group By clauses work seamlessly without having to cast columns to the right datatype.

This is an example of this same mashup without schema information:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/
                ../schemas/EMMLPrestoSpec.xsd'
    xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
    xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
    xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
    name='noSchema'>

 <output name='result' type='document' />

 <variable name="stocks" type="document" stream="true"/>

 <directinvoke  method='GET' stream='true' outputvariable='stocks'
         endpoint='http://mdc.jackbe.com/downloads/presto/data/stocks.xm

 <raql outputvariable='result'>
  select symbol, "date", open, close, volume from stocks
  where extract_year("date") = 2011 order by close
 </raql>
</mashup>
```

```
<symbol>FSLR</symbol>
<date>Mon Apr 11 00:00:00 EDT 2011</date>
<open>148.21</open>
<close>137.64</close>
<volume>1765100.0</volume>
</record>
<record>
<symbol>FSLR</symbol>
<date>Mon Apr 25 00:00:00 EDT 2011</date>
<open>137.59</open>
<close>139.57</close>
<volume>1906300.0</volume>
</record>
<record>
<symbol>FSLR</symbol>
<date>Mon Mar 07 00:00:00 EST 2011</date>
<open>147.45</open>
<close>139.74</close>
<volume>1587900.0</volume>
</record>
<record>
<symbol>F</symbol>
<date>Mon Mar 07 00:00:00 EST 2011</date>
<open>14.34</open>
<close>14.36</close>
<volume>9.19023E7</volume>
```

Sorting is defined on a numeric column, but because no datatype information is available from the original XML source the sort order in the result is wrong. But run the same query with schema information now available and the results are now sorted correctly:

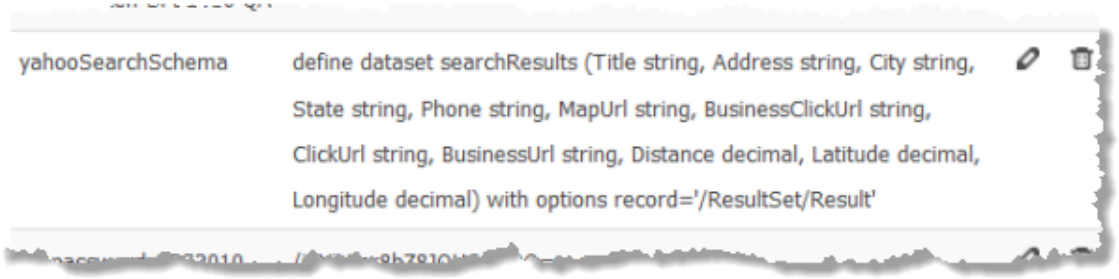
```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
        xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../s
        xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
        xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
        xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
        name='noSchema'>

  <output name='result' type='document' />
  <variable name='stockType' type='schema'>
    define dataset stocks(symbol string,
      date datetime,
      open double,
      close double,
      high double,
      low double,
      adjclose double,
      volume double)
  </variable>
  <variable name='stocks' type='variable:stockType' stream='true'/>

  <directinvoke endpoint='http://mdc.jackbe.com/prestodocs/data/stocks.xml'
                method='GET' stream='true' outputvariable='stocks' />

  <raql stream='false' outputvariable='result'>
    select symbol, "date", open, close, volume from stocks
    where extract_year("date") = 2011 order by close
  </raql>
```

```
<?xml version="1.0" encoding="UT
<records>
<record>
   <symbol>C</symbol>
   <date>Sun Apr 10 21:00:00 PDT
   <open>4.58</open>
   <close>4.42</close>
   <volume>3.81727E7</volume>
</record>
<record>
   <symbol>C</symbol>
   <date>Sun Mar 27 21:00:00 PDT
   <open>4.46</open>
   <close>4.45</close>
   <volume>2.83453E7</volume>
</record>
<record>
   <symbol>C</symbol>
   <date>Sun Mar 20 21:00:00 PDT
   <open>4.61</open>
   <close>4.46</close>
   <volume>4.68871E7</volume>
```

Global Dataset Schemas as MashZone NextGen Attributes

If a dataset will be used in many RAQL queries, you can define a schema for the dataset as a MashZone NextGen global attribute that can be easily used in different mashups.

MashZone NextGen administrators can create global attributes in the Admin Console. For dataset schemas, the value of the MashZone NextGen global attribute is the full definition of the schema. In the following example:



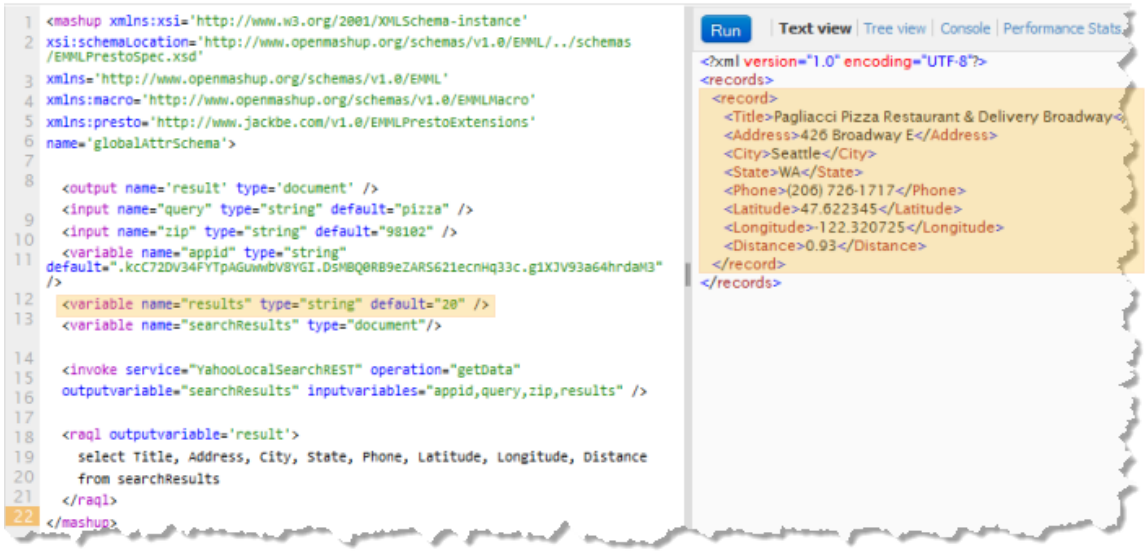
The attribute name is `yahooSearchSchema` and the full definition of the schema is a single string as the attribute value.

Once you have the dataset schema defined as MashZone NextGen global attribute you can use it in a mashup in a the `<variable>` statement with a name in the form `global.`*attribute-name* and a type of `schema`. This allows the mashup to use the global attribute to supply the schema definition. The following example retrieves the schema defined above from the MashZone NextGen global attribute named `yahooSearchSchema`:

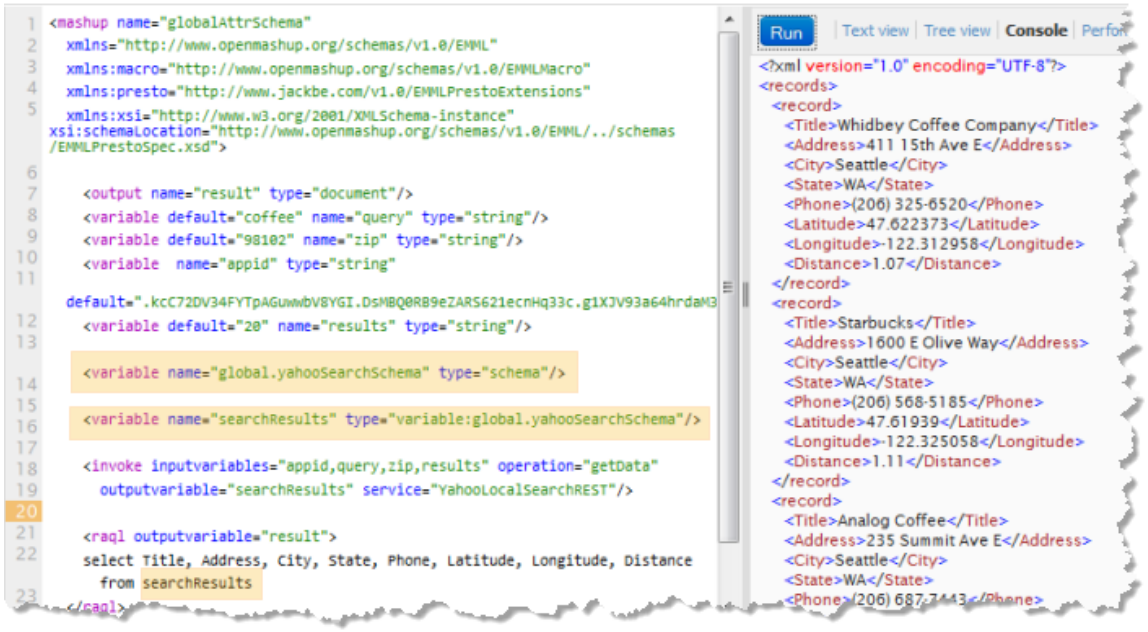
```
<mashup name="globalAttrSchema"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/
                                    EMML/../schemas/EMMLPrestoSpec.xsd">
  <output name="result" type="document"/>
  <variable default="coffee" name="query" type="string"/>
  <variable default="98102" name="zip" type="string"/>
  <variable name="appid" type="string"
default=".kcC72DV34FYTpAGuwwbV8YGI.DsMBQ0RB9eZARS621ecnHq33c.g1XJV93a64hrdaM3" />
  <variable default="20" name="results" type="string"/>
  <variable name="global.yahooSearchSchema" type="schema"/>
  <variable name="searchResults" type="variable:global.yahooSearchSchema"/>
  <invoke inputvariables="appid,query,zip,results" operation="getData"
      outputvariable="searchResults" service="YahooLocalSearchREST"/>
  <raql outputvariable="result">
    select Title, Address, City, State, Phone, Latitude, Longitude,
           Distance
    from searchResults
  </raql>
</mashup>
```

Then add the variable to hold the dataset and *reference* the schema variable using a `type` of `variable:global.`*attribute-name*, that will hold the dataset once it is loaded. The `type` identifies the named variable containing the schema for this dataset, In this example, the variable `searchResults` has a type that pulls in the `global.yahooSearchSchema` global attribute containing the schema definition.

In this example query when no schema is used, the mashup shows results of a single row even though the query to Yahoo Local Search asked for up to 20 results:



When the schema is added, supplying specific path information to rows in Yahoo's results, the query now retrieves all 20 results:



Valid Names for Datasets, Columns, Aliases, Paths and Functions

Column, Path, Alias and Function Names

Both RAQL and EMML have the following rules for valid column or alias names and the names within paths to dataset rows:

- Column, alias, path and function names may contain letters, numbers and the underscore (_) character. They *cannot* contain:
  - Spaces
  - Common punctuation or special characters such as `$`, `.` (period), `-` (dash) or `:` (colon).
- There are two exceptions to rule for using periods in names:
  - Function names use a period to separate the library name for the function from the function name. For example:

    ```
    myLibrary.myFundtion(someColumn)
    ```

  - Dataset names can be prepended to column names with a period to clarify the exact context for a column in queries where multiple datasets are used. For example:

    ```
    select datasetA.columnAA, datasetB.columnBA, ...
    ```

    See "Joins and Other Multiple-Dataset Operations" on page 760 for examples.
- Column, alias or function names *must* begin with a letter. A column name of `2000` is not a valid name, but `Y2000` is.
- The names in paths used with XML datasets must follow XML name rules. They can contain letters, numbers and underscores(_).
- Column and alias names are case insensitive and must be unique within a dataset. For example, a dataset row could not contain both a column named `item` and another named `Item`.
- Column, alias, function and path names cannot be the same as any "RAQL Reserved Keywords" on page 729.
- Functions for RAQL queries are written in Java and thus must also follow Java name requirements.

There are work arounds to handle some invalid name issues. See "Fixing Invalid Names" on page 729 for instructions.

Dataset Names

Names for datasets are the names of the EMML variables that hold the dataset stream within a mashup. Dataset names have the same character and reserved keyword restrictions as column, alias and path names (See "Column, Path, Alias and Function Names" on page 727 and "RAQL Reserved Keywords" on page 729).

They must also be unique within the mashup, but dataset names *are* case sensitive. Although it is not a good practice, a mashup could have a dataset named `events` and another named `EVENTS`.

Fixing Invalid Names

RAQL automatically fixes some issues in invalid names for datasets. With CSV datasets, for example, RAQL will replace spaces in column names with an underscore (_) or change numeric column names to `column_`*original-number* .

You can also enclose invalid names in double quote marks (") to fix many invalid name problems.

Note: You *cannot* use single quote marks (') to enclose invalid names because this indicates the string is a literal value rather than a name. See "Literal Values in Conditions or Expressions" on page 739 for more information.

For example, this query will run even though three of the columns names are invalid or match reserved keywords:

```
select "dash-column", "dot.column", "date", valid_column from myDataset
```

RAQL Reserved Keywords

The following SQL keywords are reserved words for RAQL queries. Keywords are shown in lower case. Matching is not case sensitive.

Column and function names in queries that match reserved keywords must be delimited with double quote marks (") to prevent errors in query execution.

Note: Single quote marks (') are used to delimit literal values that are string, and thus are not valid to delimit names. See "Literal Values in Conditions or Expressions" on page 739 for more information.

abs	double	min	slide
all	duration	minus	sqrts
and	else	minute	stddev
any	end	minutes	stddev_pop
array	except	mod	stddev_samp
as	exists	month	string
asc	exp	months	struct
avg	false	natural	sum

between	float	not	then
big_decimal	floor	now	time
blob	following	null	timestamp
boolean	from	on	to
by	full	or	true
byte	group	order	trunc
case	having	outer	truncate
cast	hour	over	unbounded
ceil	hours	partition	union
ceiling	in	pattern	unknown
character	inner	power	using
character_length	integer	preceding	variance
char_length	intersect	range	var_pop
clob	is	recursive	var_samp
count	join	ref	week
cross	left	relative	weeks
current	like	right	when
date	limit	round	where
day	ln	row	window
days	long	rows	with
default	matching	seconds	within
define	max	seconds	year

desc	measures	select	years
distinct	millisecond	shift	
do	milliseconds	short	

From Subqueries

You can use subqueries in RAQL in the From clause *only*. The following example has three levels of subqueries based on the stocks dataset introduced in "Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics" on page 696 in Getting Started.

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/
   ../schemas/EMMLPrestoSpec.xsd'
   xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
   xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
   xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
   name='subquery'>

   <output name='result' type='document' />
   <variable name="stocks" type="document" stream="true"/>

   <directinvoke  method='GET' stream='true' outputvariable='stocks'
      endpoint='http://mdc.jackbe.com/downloads/presto/data/stocks.xml'/>

   <raql outputvariable='result'>
      select yr, qtr, pct_change
      from (
          select yr, qtr,
          (this_qtr - prev_qtr)/number(nvl(prev_qtr,this_qtr)) * 100 as pct_change
          from (
              select yr, qtr, this_qtr, lag(this_qtr) over () as prev_qtr
              from (
                  select extract_year("date") as yr, quarter("date") as qtr,
                         decimal(mean(volume)) as this_qtr
                  from stocks
                  group by extract_year("date"), quarter("date")
                  order by yr, qtr
                 )
              )
          )
      where pct_change > 15
   </raql>
</mashup>
```

Each level within the query builds on the previous level:

1. The innermost subquery groups stocks data by year and quarter and calculates the average volume for each group.
2. This average volume is used in the inner middle subquery to retrieve the previous quarter's average volume for each row.

3. These two average volumes are then used in the outer middle subquery to calculate the percentage of change using a simple math expression.
4. The final outer query then filters the results to include only those rows where the percentage of change is greater than 15%.

Also of interest in the inner middle subquery is the use of the `lag()` analytical function to find and add the previous quarter's average volume to each row. In the outer middel subquery, the `nvl()` plain function in the equation for the percentage of change handles the first row where the previous quarter value is `null`.

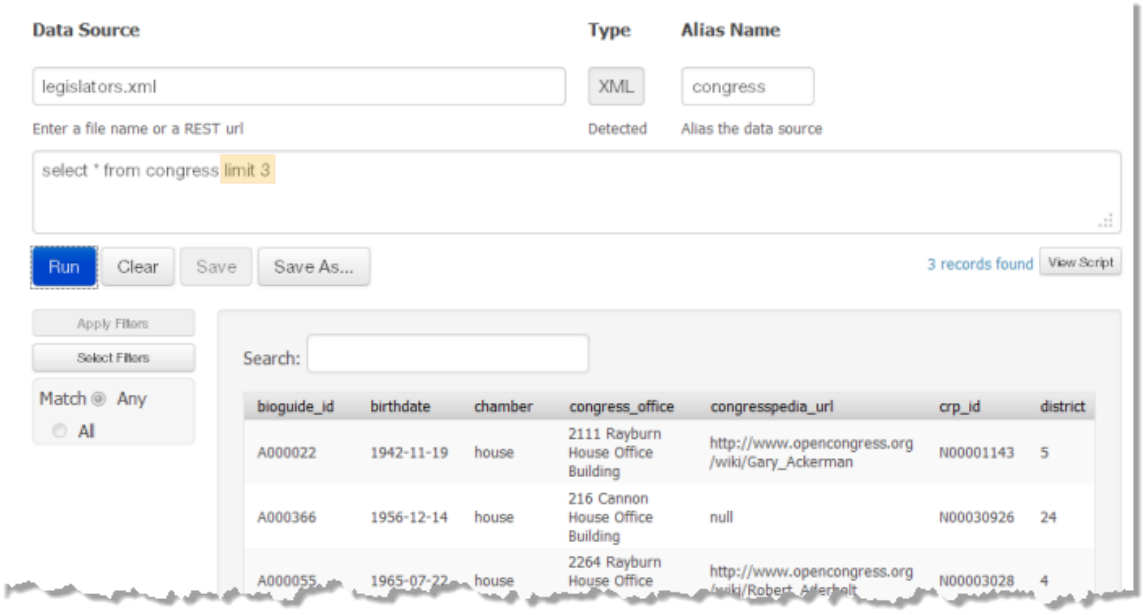
Use of Aliases and Subqueries

The example shown above also illustrates the use of subqueries to allow the use of aliases in calculations in the Select clause or in conditions in the Where clause. The inner middle subquery defines `prev_qtr` as an alias. This alias is then used in the subquery one level up in the calculation of the `pct_change` column.

Similarly, the use of `pct_change` in the Where clause can only occur in the final outer query because `pct_change` has been defined in the previous subquery. If the query attempted to use this in Where in the intermediate query, it would fail with an error.

Limit the Rows Returned

You can use the `limit` keyword at the end of RAQL queries to limit the number of rows to return. One common use is to do an initial query that retrieves all columns, but a small number of rows to get a look that the dataset you are working with. For example:



Where Complex Conditions

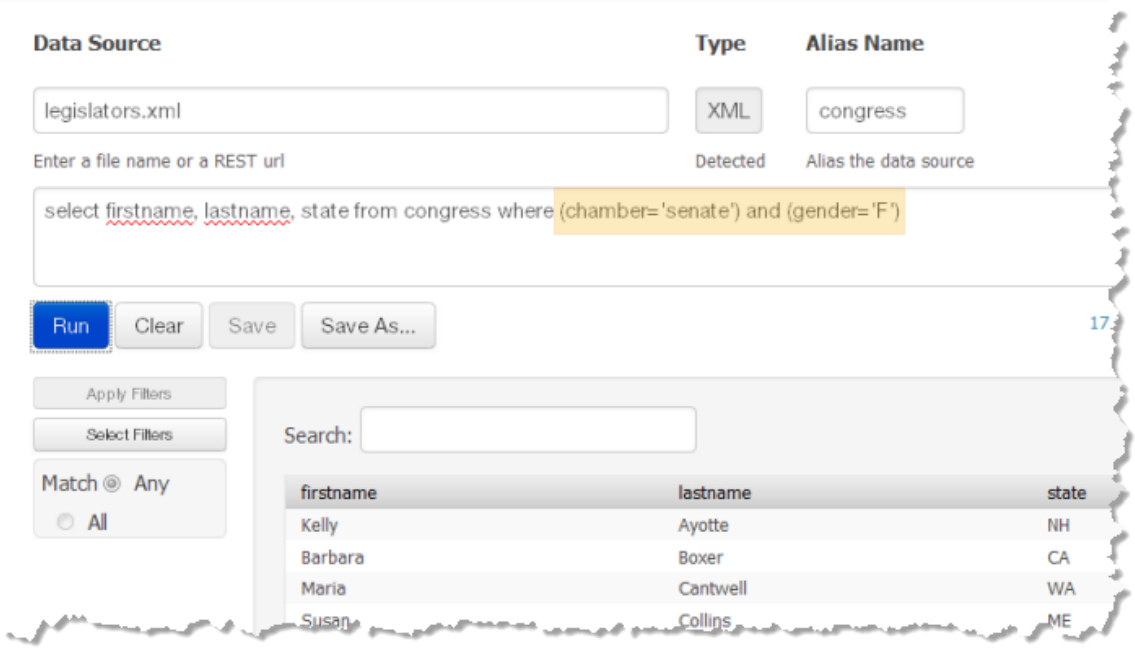
Where clauses can use logical operators, comparison operators or arithmetic operators combined with parenthese to define complex conditions for queries. See "RAQL Operators" on page 773 for a complete list of valid operators for Where clauses.

This topic provides examples of many of the common techniques you can use:

- Multiple Required Conditions
- Combining Logical Conditions
- Comparing Dates or Numbers
- Calculations in Conditions
- Negative Comparisons

Multiple Required Conditions

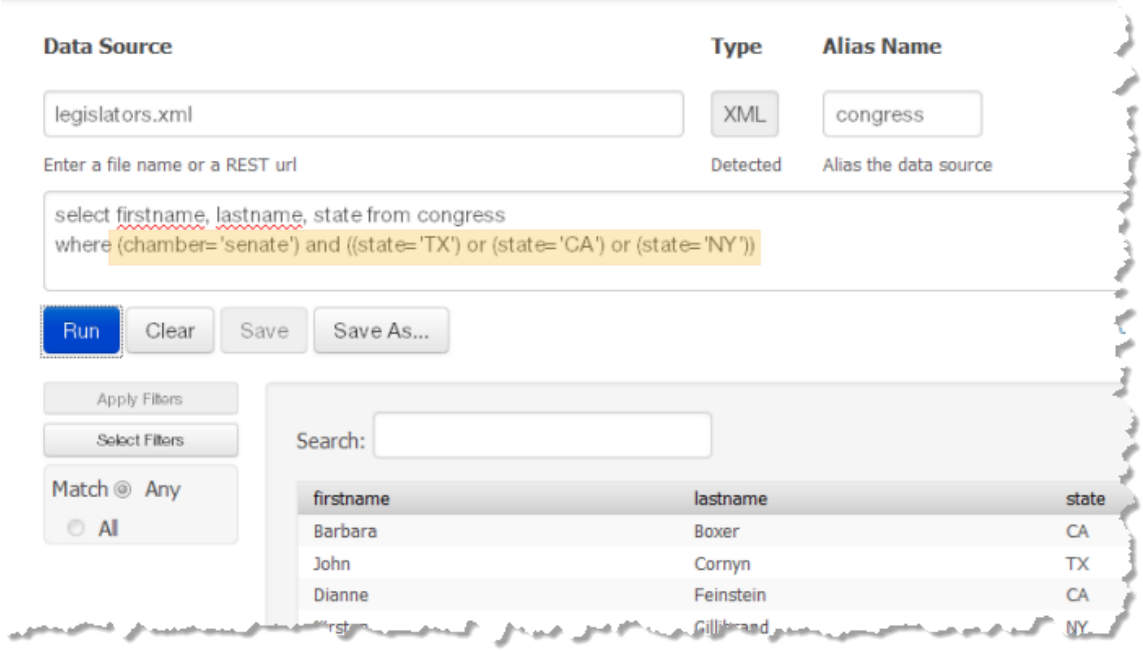
Use the `and` logical operator to combine conditions when all are required:



Combining Logical Conditions

Use the `or` logical operator to define conditions where multiple matches are allowed. You can also combine `and` and `or` to define more complex conditions. Use parenthese to indicate the appropriate precedence.

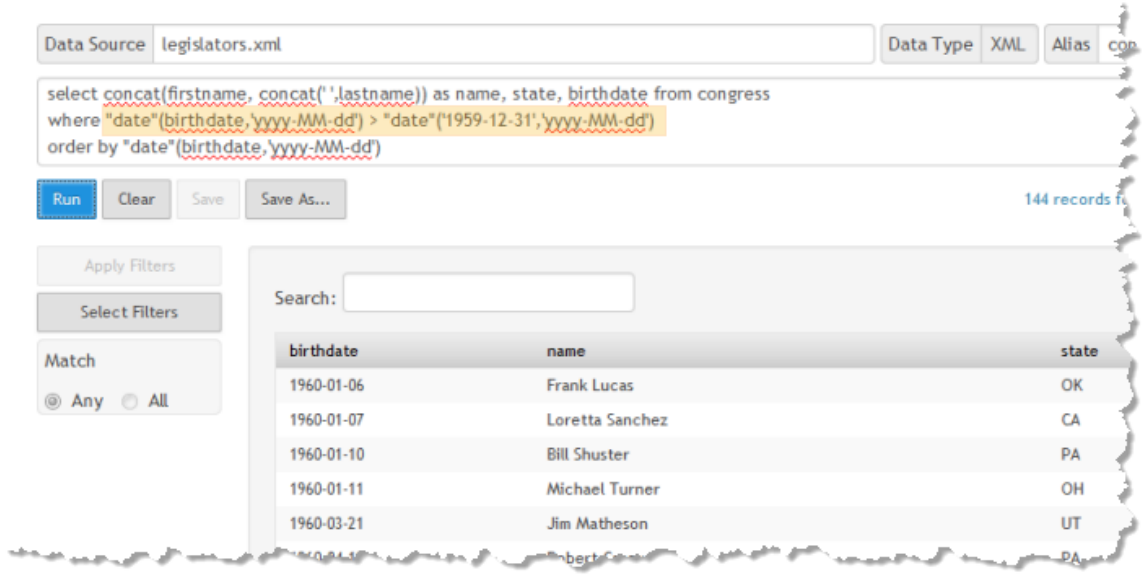
This example selects senators from Texas, New York or California:



Comparing Dates or Numbers

You can use the common math comparison operators to compare numbers or dates. In many cases, you also need to cast column data to an appropriate datatype for the comparison to be successful.

This example selects legislators whose birthdate is January 1, 1960 or later:

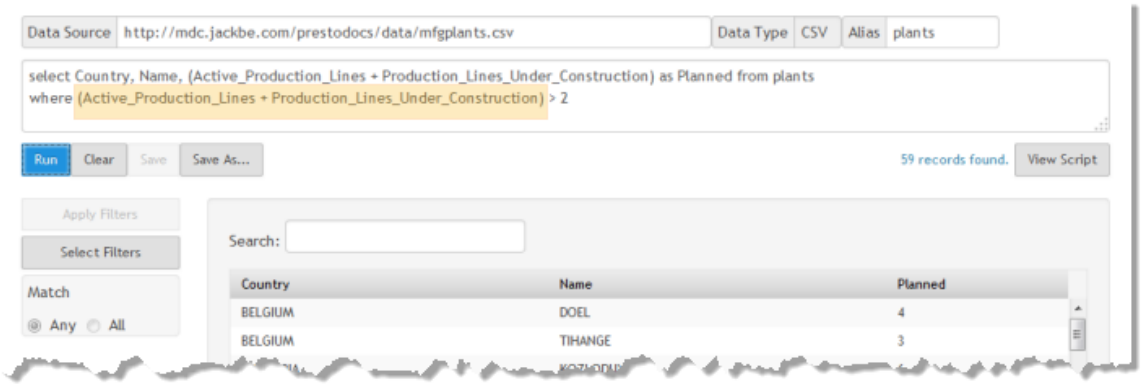


See also "Literal Values in Conditions or Expressions" on page 739 for more information.

Calculations in Conditions

You can perform common arithmetic operations within the Where clause. Use parenthese to resolve any precedence issues with math operators.

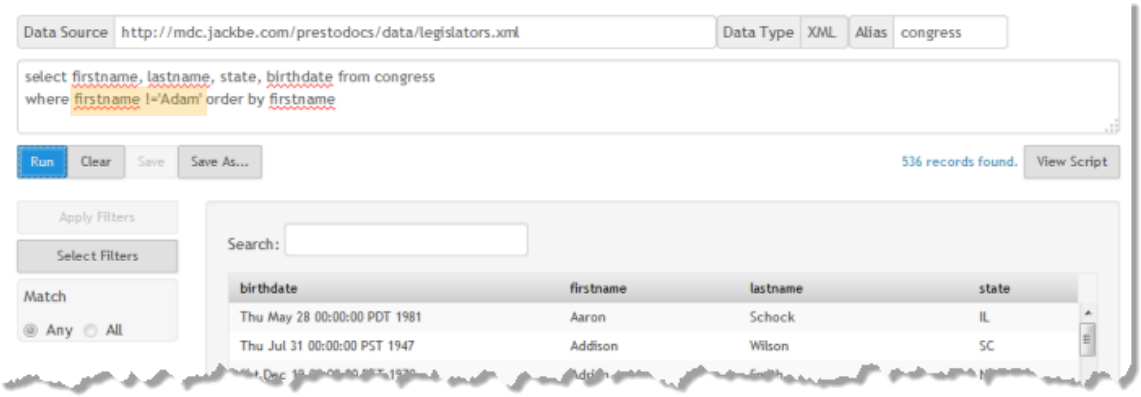
This example selects plants based on the sum of currently active plants and plants under construction:



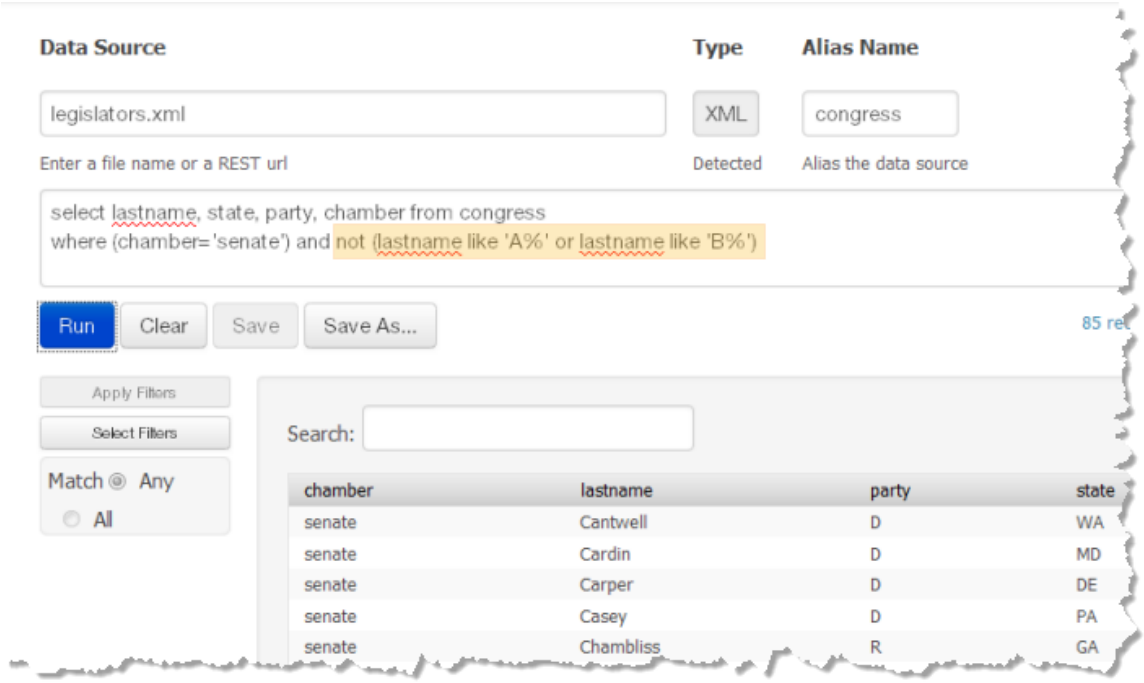
Negative Comparisons

For negative comparison, you can use the `!=` comparison operator or the `not` logical operator.

This example selects legislators whose birthdate is January 1, 1960 or later and whose first name is not `Adam`:



In many cases, however, the `not` logical operator is more flexible. This example query finds all senators whose last name does not begin with A or B:

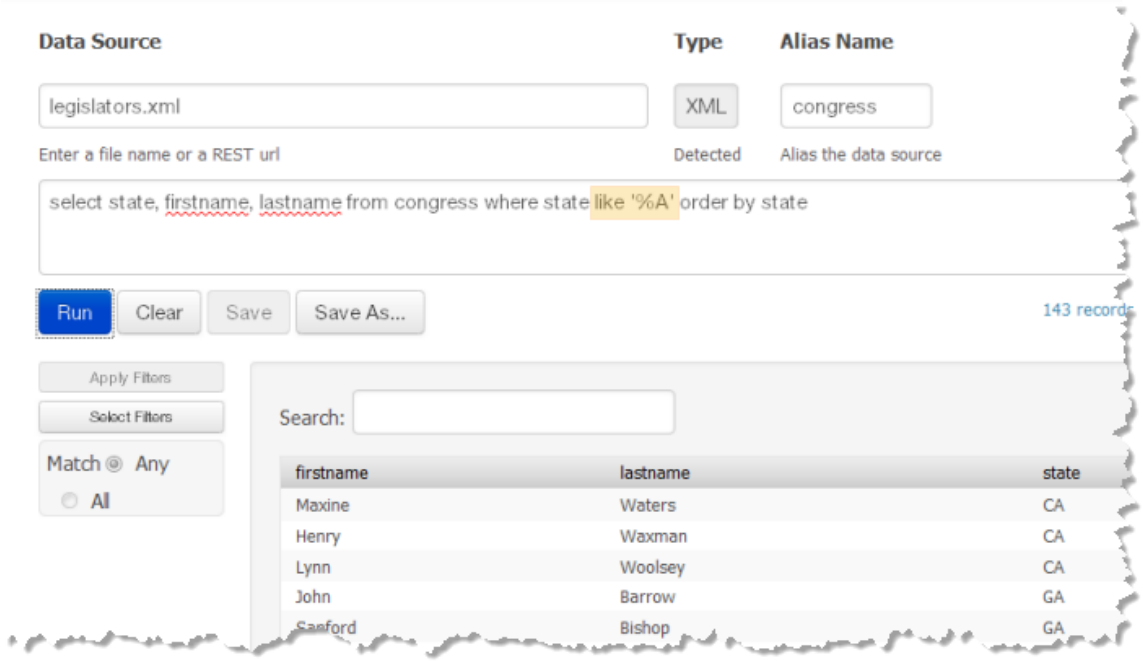


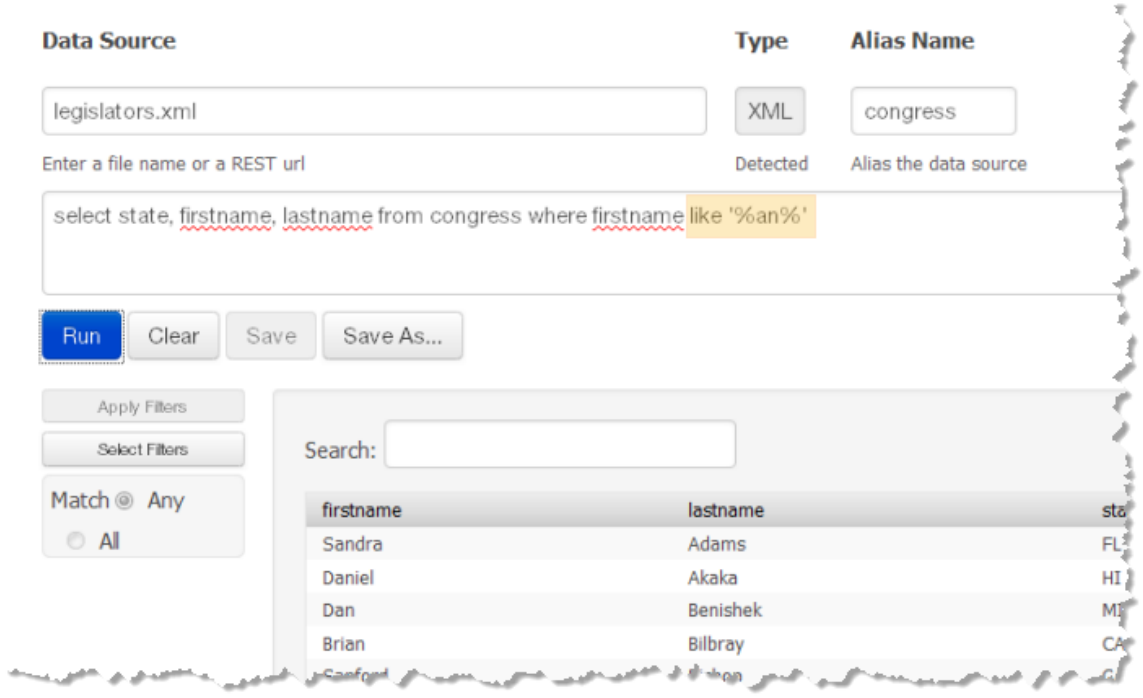
Where Text Like Patterns

You can define filter conditions in the Where clause for string columns based on a simple matching pattern using the `like` *pattern* keyword. The Like pattern uses the `%` symbol as a wildcard to represent zero to any number of characters and the _ symbol to represent a single character.

You can find an example of Like with a pattern using wildcards at the end in Group and Analyze Rows with Row Detail. This selects rows based on the stock symbol starting with either `D%` or `N%`.

You can also define patterns to match the end or middle of the string. For example:



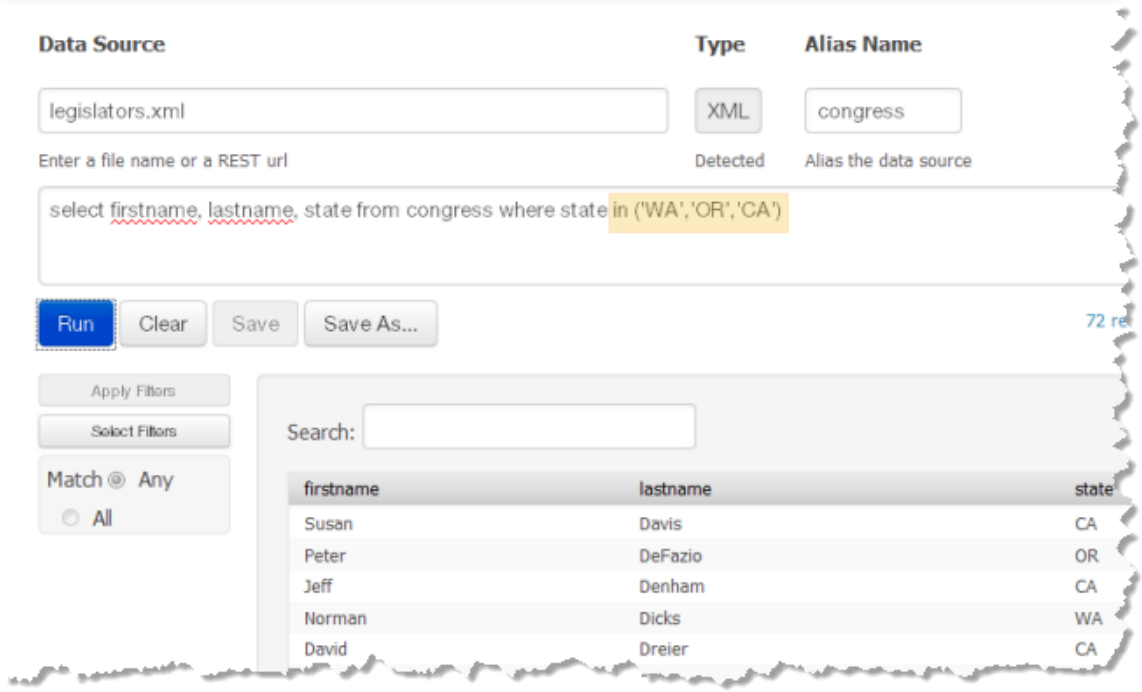


Where in Sets

You can define filter conditions for string columns to match rows based on an enumerated set of values using the `in ('value1', 'value2'[,...])` keyword and set definition.

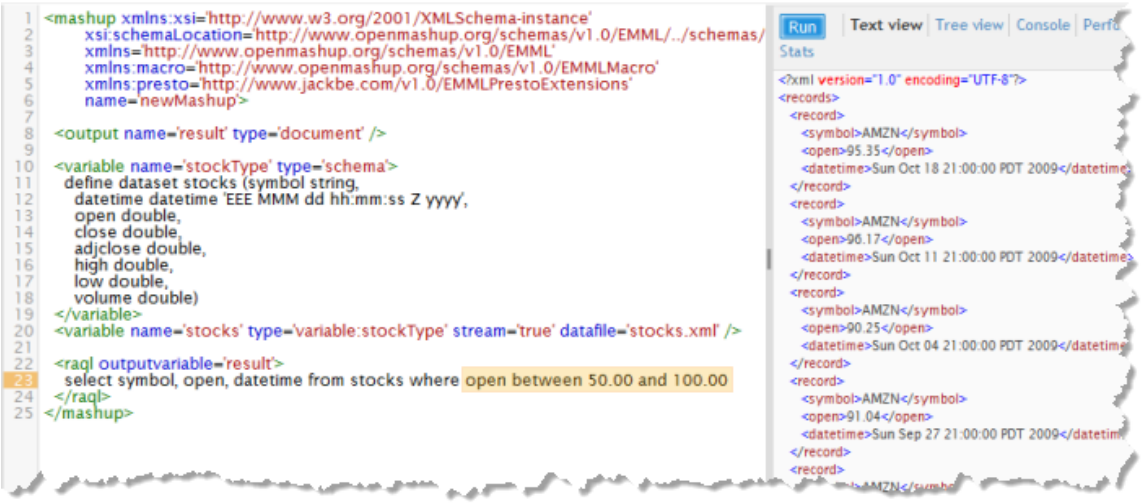
Note: The set must be defined inside *parentheses*, rather than the brackets typically used in SQL.

For example:



Where Between a Range

Between conditions simplify Where clauses for numeric columns defining a range of values to include in results. For example:



Parameters in Where Clauses

RAQL does *not* currently support the :*parm-name* syntax in Where clauses to supply values for filter conditions from named parameters. You can, however, make RAQL queries dynamic by:

- Supplying parameter values in Where clauses using Dynamic Mashup Expressions. This uses an EMML syntax in the form:

  {$*parameter-name* }

  To refer to input parameters or other variables defined in the mashup.

- Building the entire query dynamically using the EMML"<assign>" on page 503 element and variables or input parameters in the mashup. This technique is useful when you need to dynamically set different aliases, functions or entire query clauses.

  See "Creating Dynamic RAQL Queries" on page 764 for an example and more information.

This example shows a simple Where clause where the threshold for a filter is supplied by an input parameter to the mashup:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/
      xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
      xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
      xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
      name='whereParameters'>

 <output name='result' type='document' />
 <input name='threshold' type='number' default='2'/>

 <variable name='plants' type='document' subtype='csv' />

 <directinvoke endpoint='http://mdc.jackbe.com/prestodocs/data/mfgplants.csv'
          method='GET' stream='true' outputvariable='plants' />

 <raql outputvariable='result'>
  select Name, Country, Active_Production_Lines from plants
  where cast(Active_Production_lines as double) &gt; {$threshold}
 </raql>
</mashup>
```

Literal Values in Conditions or Expressions

Conditions that you use in Where or Having clauses, as well as other expressions you can use in functions, can contain literal values. To have literal values recognized correctly, you must follow some basic rules based on the type of data for that literal:

- *String*: literal values must be enclosed in single quote marks (') to distinguish them as a literal value rather than a query keyword or the name of a column or dataset. For example:

```
select lastname,state from congress where state = 'TX'
```

Note: You cannot use double quotes (") around literal strings as this indicates the string is a name instead. See "Valid Names for Datasets, Columns, Aliases, Paths and Functions" on page 727 for more information.

- *Numeric*: literal values do not require any delimiters. Include the decimal point or sign characters, but other punctuation such as thousand separators is not required. For example:

```
select item,qty,price from invoices where price < 50.00
```

- *Date or Time*: literal values usually require a casting function to ensure that they are properly recognized. For example:

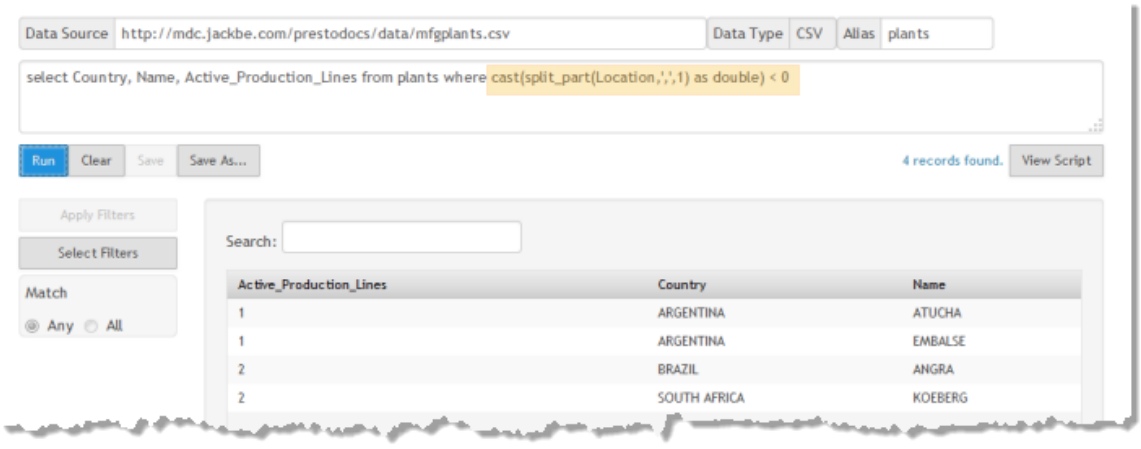
```
select lastname, state, birthdate from congress
where birthdate <
to_date('1959/12/31','YYYY/MM/dd')
```

Plain Functions in Where

Plain functions can be used in any RAQL query clause. You may use any MashZone NextGen built-in plain function (see "Built-In RAQL Functions" on page 775 for a list) or user-defined plain functions that you or other MashZone NextGen developers have added.

You can find a simple example of using plain functions in a Where clause in Use Plain Functions to Update, Select or Sort Rows. See also "Comparing Dates or Numbers" on page 734 for an example of using casting functions in Where clauses.

This example query uses nested functions to select manufacturing plants that are in the southern hemisphere, by comparing the latitude in the data:

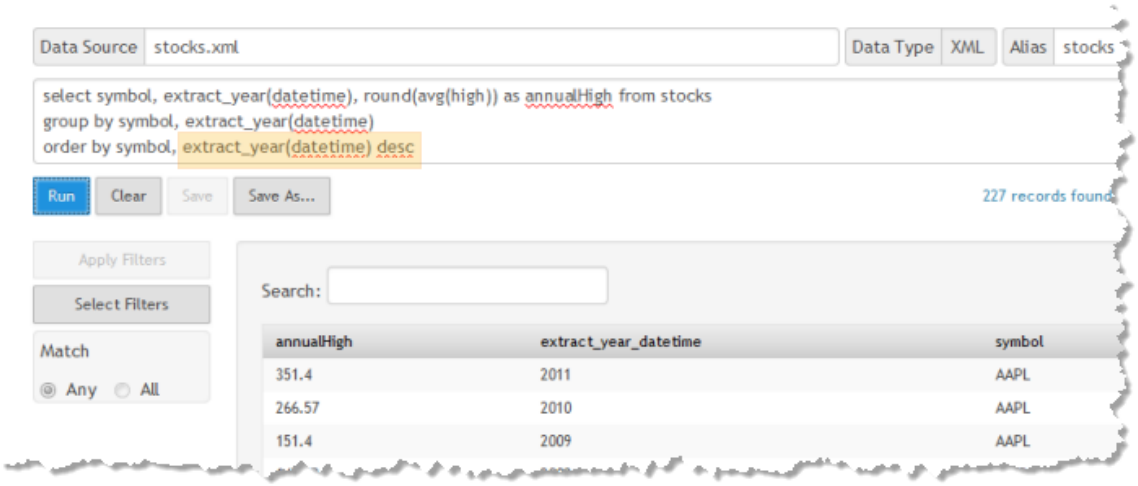


Sort Directions and Multi-Level Sorts

Sort Directions

By default, rows included in query results are sorted in ascending order based on the column expressions in the Order By clause. You can change the sort order for specific columns by using the `desc` or `asc` keywords in the column expression.

For example:



Character Sets and Sort Collations

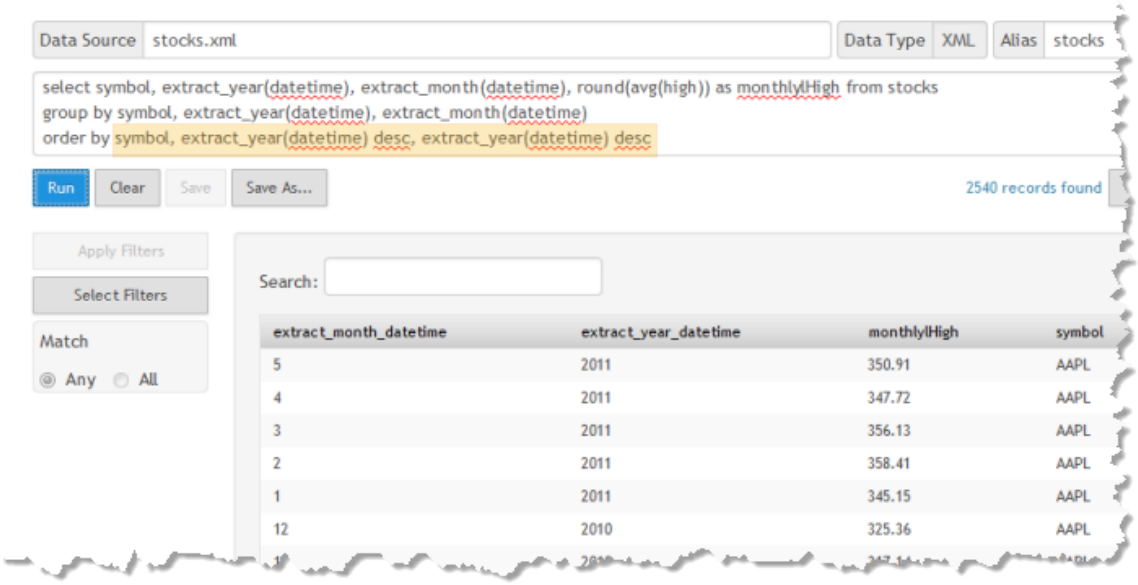
For string values, sorting depends on both the character set of the text and the *sort collation* defined for that character set. For example, many European languages have upper-case and lower-case letters which are sorted separately. Many Asian languages use ideographs where this concept does not apply for sorting.

Both the character sets that RAQL can work with and the collations used for string sorting are defined by the version of Java used by the application server that MashZone NextGen is hosted in.

Multi-Level Sorts

Multiple levels of sorting are defined simply by the columns included in the Order By column expression. Columns are listed in the order for sorting separated by commas.

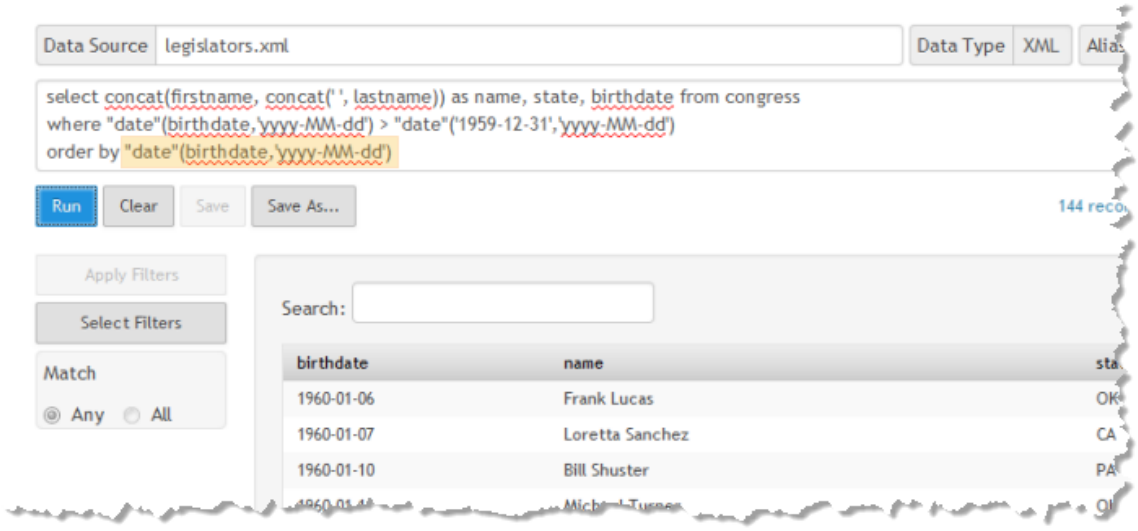
A simple, two-column sort is shown in Group and Analyze Rows in Getting Started. There is no specific limit for sorting levels, however. This example shows a three-level sort:



Plain Functions in Sorts

Plain functions can be used in any RAQL query clause. You may use any MashZone NextGen built-in plain function (see "Built-In RAQL Functions" on page 775 for a list) or user-defined plain functions that you or other MashZone NextGen developers have added.

A simple example is shown here to sort by a column with dates:



Other examples of using plain functions in Order By clauses are shown in:

- Sort Directions and Multi-Level Sorts
- Use Plain Functions to Update, Select or Sort Rows

Multi-Level Group Calculations

The example of a Group By clause in Getting Started, "Group and Analyze Rows" on page 699, used a composite group containing two group columns to segment stock data into sets by symbol and then year within symbol.

Once groups are defined, you determine what analysis occurs and is returned using analytic functions in the Select clause. Each group returns one row of data, so the functions you choose must be aggregate analytic functions that return a single value for all rows in the group. See "Built-In Analytic Functions: Aggregate and Window" on page 785 for the list of built-in MashZone NextGen analytic functions that you may use with Group By clauses.

For more examples, see:

- "Example 111. Single Level Group" on page 743
- "Example 112. Three Group Levels" on page 743
- "Example 113. Groups Using an Aggregate Analytic Function" on page 744
- "Example 114. Aggregate on Multiple Groups with ROLLUP, CUBE and GROUPING SETS" on page 745
- "Example 115. Special Grouping Sets: ROLLUP and CUBE" on page 747
- "Example 116. Using composite group criteria" on page 750
- "Example 117. Null values in Grouping Set aggregations" on page 750
- "Example 118. Combining and Merging Grouping Sets" on page 753

Single Level Group

You can use a single column group such as this example which counts the number of legislators in each house of the US Congress:

```
SELECT chamber, COUNT(chamber) FROM congress GROUP BY chamber
```

Chamber	COUNT_chamber
senate	100
house	439

Three Group Levels

With each level of group, you add columns to the expression. This example groups stock data by symbol, year and quarter within year to derive the average low price for each quarter:

```
SELECT symbol,extract_year(datetime), quarter(datetime), round(avg(low)) AS qtrLow
  FROM stocks
```

```
GROUP BY symbol,extract_year(datetime), quarter(datetime)
ORDER BY symbol,extract_year(datetime) DESC, quarter(datetime)
```

Symbol	Extract_year_datetime	Quarter_datetime	QtrLow
AAPL	2011	1	337.0
AAPL	2011	2	335.0
AAPL	2010	1	206.0
AAPL	2010	2	242.0
AAPL	2010	3	253.0
AAPL	2010	4	305.0
AAPL	2009	1	89.0
AAPL	2009	2	126.0
AAPL	2009	3	161.0
AAPL	2009	4	192.0
AAPL	2008	1	130.0
AAPL	2008	2	168.0
...	...	...	...

The columns that appear in the Select clause must either be used for group in the Group By clause or be used in aggregate calculations. It is also quite common to sort by the columns used in grouping.

Groups Using an Aggregate Analytic Function

This example calculates the standard deviation, using the `stddev()` aggregate analytic function, for the high prices of each stock and year:

```
SELECT symbol, extract_year(datetime) AS yr, stddev(high) AS highStdDev
  FROM stocks
GROUP BY symbol, extract_year(datetime)
ORDER BY symbol, yr
```

Symbol	Yr	HighStdDev
AAPL	1984	1.6057616847575154
AAPL	1985	4.628989380498321
AAPL	1986	5.626270136472927
AAPL	1987	14.69298548409742
AAPL	1988	2.4555231784570233
AAPL	1989	4.2820135239165875
AAPL	1990	4.529251236509766
AAPL	1991	7.4271879581244775
AAPL	1992	7.62725912432657
AAPL	1993	13.459208620188244
AAPL	1994	4.371574862654053
AAPL	1995	4.069107853754948
AAPL	1996	3.336661126725468
...	...	...

The columns that appear in the Select clause must either be used for group in the Group By clause or be used in aggregate calculations. It is also quite common to sort by the columns used in grouping.

Aggregate on Multiple Groups with ROLLUP, CUBE and GROUPING SETS

The examples above partitioned the data and calculated an aggregate value for each partition based only on a single group criterion. However, let's assume you are not only interested in the standard deviation of high prices for each stock and year, but additionally would like to see the overall standard deviation of high prices per stock, and the total standard deviation of high prices over all stocks. You could run three separate GROUP BY queries and combine their results as follows:

```
SELECT symbol, extract_year(datetime) AS yr, stddev(high) AS highStdDev FROM stocks
```

```
       GROUP BY symbol, extract_year datetime)
  UNION ALL
SELECT symbol, CAST(NULL AS DATE) AS yr, stddev(high) AS highStdDev FROM stocks
       GROUP BY symbol
  UNION ALL
SELECT CAST(NULL AS STRING) AS symbol, CAST(NULL AS DATE) AS yr, stddev(high) AS
       highStdDev FROM stocks ORDER BY symbol NULLS FIRST, yr DESC NULLS FIRST
-- we specify an ordering here to see the aggregated symbol and year columns
in the example output --
```

MashZone NextGenfully supports the SQL syntax for complex GROUPING SETS, which allows to specify a single set of group expressions in the group-by clause, and to specify multiple sets that the data is partitioned and aggregated upon. Using this syntax, the above query could easily be rewritten to:

```
SELECT symbol, extract_year(datetime) AS yr, stddev(high) AS highStdDev FROM stocks
  GROUP BY GROUPING SETS ( (symbol, extract_year(datetime)), symbol, () )
  ORDER BY symbol NULLS FIRST, yr DESC NULLS FIRST
-- we specify an ordering here to see the aggregated symbol and year columns
in the example output --
```

Note: Please note, that this query is not only much shorter, but may also be evaluated more efficiently by the query engine.

The above GROUPING SETS clause defines three sets of group expressions (`symbol, extract_year(datetime)`), `symbol`, and the `empty grouping set ()`, so that in the result of the query we will find a row for every unique pair of (`symbol, extract_year(datetime)`) values, concatenated by rows for each unique value of `symbol`, and a single row representing the empty group. For each such group row, the values of the group expressions that are not considered in the current grouping are filled with `NULL`, e. g. the `yr` column is `NULL` for the second grouping by `symbol` (see also section "Example 117. Null values in Grouping Set aggregations" on page 750).

Symbol	Yr	HighStdDev	
		95.99022287735269	This row indicates the HighStdDev aggregated over all stocks and all years.
AAPL		66.38352453765187	This row indicates the HighStdDev for symbol AAPL aggregated over all years.
AAPL	2011	7.78515914570644	
AAPL	2010	37.77321347891091	
AAPL	2009	40.274807396460886	
AAPL	2008	33.6190395688354	
AAPL	2007	37.83959132285136	

Symbol	Yr	HighStdDev
AAPL	2006	9.55659436868603
AAPL	2005	15.564417219126476
AAPL	2004	13.782231301956635
AAPL	2003	3.4495406114241156
AAPL	2002	4.4393911367617065
AAPL	2001	2.6334977228792624
AAPL	2000	43.679940128685516
...	...	...

Special Grouping Sets: ROLLUP and CUBE

It is very common, to have hierarchical dimensions on which to partition the data. For instance, year and month could be such a dimension hierarchy. If you are interested in the highest price values among all stocks for each dimension level, you could use the GROUPING SETS syntax to specify the following four sets of group expressions:

```
SELECT extract_year(datetime) AS y, extract_month(datetime) AS m,
       max(high) AS maxHigh
  FROM stocks
GROUP BY GROUPING SETS (  (extract_year(datetime), extract_month(datetime)),
                          (extract_year(datetime)),
                          ()  )
ORDER BY y NULLS FIRST, m NULLS FIRST
-- we specify an ordering here to see the aggregated symbol and year columns
in the example output --
```

However, for queries like this, where the grouping set dimensionality is reduced from right to left, MashZone NextGen supports the SQL keyword `ROLLUP` , which - given a list of group expressions - rolls-up the list by consecutively removing one expression from the end of the list and calculating the next higher level aggregate, up to the overall aggregate.

Hence, the above query is equivalent to the following query using the `ROLLUP` syntax instead:

```
SELECT extract_year(datetime) AS y, extract_month(datetime) AS m, max(high) AS maxHigh
 FROM stocks
GROUP BY ROLLUP (extract_year(datetime), extract_month(datetime))
ORDER BY y NULLS FIRST, m NULLS FIRST
-- we specify an ordering here to see the aggregated symbol and year columns
in the example output --
```

Note: Please note, that the roll-up of the grouping columns is reflected in the result by a `NULL` value in the corresponding group column (shown as an empty column).

Y	M	MaxHigh	
		747.24	This row indicates the MaxHigh aggregated over all years and all months.
1984		128.5	This row indicates the HighStdDev for year 1984 aggregated over all months.
1984	9	128.5	
1984	10	127.75	
1984	11	127.75	
1984	12	124.5	
1985		158.75	This row indicates the HighStdDev for year 1985 aggregated over all months.
1985	1	137.63	
1985	2	138.25	
1985	3	136.38	
1985	4	130.38	
1985	5	133.25	
1985	6	130.63	
1985	7	132.75	
...	...	...	...

In addition to the `ROLLUP` keyword, which enumerates a hierarchy of grouping sets,MashZone NextGensupports the SQL keyword `CUBE`, to enumerate the power set of grouping sets, that is the set of all subsets. As an example, consider the group

expressions chamber, state and gender. Then the clause `CUBE(party, state, gender)` translates into the following grouping sets:

```
(party, state, gender)
(party, state)
(party, gender)
(state, gender)
(party)
(state)
(gender)
()
```

In comparison, these would be the grouping sets produced by `ROLLUP(party, state, gender)`:

```
(party, state, gender)
(party, state)
(party)
()
```

That is the query:

```
SELECT party, state, gender, count(*) FROM congress
 WHERE chamber = 'house' AND state LIKE 'K%'  -- only to limit the result size here --
 GROUP BY GROUPING SETS (  (party, state, gender),
                           (party, state), (party, gender), (state, gender)
                           (party), (state), (gender),
                            ()  )
```

It yields the exact same results as the much shorter query:

```
SELECT party, state, gender, count(*) FROM congress
 WHERE chamber = 'house' AND state LIKE 'K%'  -- only to limit the result size here --
 GROUP BY CUBE(party, state, gender)
 ORDER BY party NULLS FIRST, state NULLS FIRST, gender NULLS FIRST
```

Part	Stat	Genc	COUNT	
			10	Overall number of congressmen representing Kansas or Kentucky
		F	1	Overall number of female congressmen representing Kansas or Kentucky
		M	9	Overall number of male congressmen representing Kansas or Kentucky
	KS		4	Number of congressmen representing Kansas
	KS	F	1	Number of female congressmen representing Kansas
	KS	M	3	Number of male congressmen representing Kansas
	KS		6	Overall number of congressmen representing Kentucky

Part	Stat	Genc	COUNT	
	KS	M	6	Number of male congressmen representing Kentucky
D			2	Number of democratic congressmen representing Kansas or Kentucky
D		M	2	Number of male democratic congressmen representing Kansas or Kentucky
D	KS		2	Number of democratic congressmen representing Kentucky
D	KS	M	2	Number of male democratic congressmen representing Kentucky
R			8	Number of republican congressmen representing Kansas or Kentucky
R		F	1	Number of female republican congressmen representing Kansas or Kentucky
...	...	...	...	...

Using composite group criteria

With the `CUBE` and `ROLLUP` syntax it is also possible, to combine multiple columns into a group expression, that is treated as if it was a single column. For instance, the expression `ROLLUP(a, (b,c), d)` translates into the grouping sets (a, (b,c), d), (a, (b,c)), (a), (). I.e., c is not rolled-up from (b,c).

Note: It is a best practice, to use `ROLLUP` and `CUBE` only in cases, when all calculated groupings are relevant. If only a subset of the produced groupings is of interest, it is more efficient to use the corresponding GROUPING SETS expression generating only those relevant groupings.

Null values in Grouping Set aggregations

In the result of a grouping query over multiple grouping sets, `NULL` values in one of the group columns may have different meanings:

1. If the group column contains `NULL` values in its source table, all those `NULL` values are treated equally and grouped into a separate `NULL` group.
2. If the group column is aggregated, i. e., it is omitted in the grouping set for this grouping's row, then the group column is also `NULL`.

Consequently, it is not possible to tell the two different meanings of `NULL` apart only from looking at the result row itself. Therefore, a special SQL function `GROUPING` is provided, the purpose of which is to tell for a single group expression, whether or not this expression has been aggregated in the result (result = 1) or not (result = 0).

```
SELECT symbol, GROUPING(symbol) AS grp_symbol, extract_year(datetime) AS yr,
               GROUPING(extract_year(datetime)) AS grp_year,
               stddev(high) AS highStdDev
  FROM stocks
GROUP BY ROLLUP (symbol, extract_year(datetime))
ORDER BY symbol NULLS FIRST, yr DESC NULLS FIRST
```

Symb	Grp_syml	Yr	Grp_ye	HighStdDev	
	1		1	95.99022287735269	NULL values here indicate that symbol and yr have been aggregated.
AAPL	0		1	66.38352453765187	NULL value in column yr here indicates that yr has been aggregated.
AAPL	0	2011	0	7.78515914570644	
AAPL	0	2010	0	37.77321347891091	
AAPL	0	2009	0	40.274807396460886	
AAPL	0	2008	0	33.6190395688354	
...	...	...	...	...	...

Additionally, this function is overloaded to also take a number of group expressions as parameters. In this case, the result will be a numerical id for each resulting grouping. In order to make this id unique among all groupings, all group expressions should be given as parameters to the function.

```
SELECT symbol, extract_year(datetime) AS yr,
       GROUPING(symbol, extract_year(datetime)) AS grp,
                stddev(high) AS highStdDev
     FROM stocks
    WHERE extract_year(datetime) BETWEEN 2009 AND 2011 -- only to limit the result set here --
    GROUP BY ROLLUP (symbol, extract_year(datetime))
    ORDER BY symbol NULLS FIRST, yr DESC NULLS FIRST
```

Symbc	Yr	Gr	HighStdDev	
		3	151.89615745136808	There is only one row in this grouping representing the overall aggregate.
AAPL		1	82.34565115212739	This row belongs to the grouping 1 which relates to the grouping set (symbol).
AAPL	20110		7.785159145706435	This row belongs to the grouping 0 which relates to the grouping set (symbol, extract_year(datetime)).
AAPL	20100		37.773213478910954	
AAPL	20090		40.274807396460936	
AAPL		1	41.95096980230709	This row also belongs to the grouping 1 which relates to the grouping set (symbol).
...	...	...	...	...

The `GROUPING(col1, … colN)` result is internally computed by combining the `GROUPING` results for each single column into a bit set which is finally interpreted as a `LONG` value. Example: Assume the following `GROUPING` values for single group columns a, b and c:

```
GROUPING(a) = 1, GROUPING(b) = 0 and GROUPING(c) = 1
```

Then `GROUPING(a,b,c)` yields the bit set 101 which is interpreted as the `LONG` value 5.

To make things clear, the following query shows all the distinct groupings resulting from a group by `CUBE` over three columns. Note how the single `GROUPING` values for each column are concatenated into a bit set representing the final `GROUPING` value.

```
SELECT DISTINCT
       GROUPING(symbol) AS grpSmbl, GROUPING(extract_year(datetime)) AS grpYr,
                GROUPING(extract_month(datetime)) AS grpMnth,
       CAST(GROUPING(symbol) AS STRING) || CAST(GROUPING(extract_year(datetime)) AS STRING) ||
            CAST(GROUPING(extract_month(datetime)) AS STRING) AS bits,
       GROUPING(symbol, extract_year(datetime), extract_month(datetime)) AS grp
  FROM stocks
GROUP BY CUBE (symbol, extract_year(datetime), extract_month(datetime))
ORDER BY grp ASC
```

grpSymbol	grpYear	grpMonth	bits	grp
0	0	0	000	0

grpSymbol	grpYear	grpMonth	bits	grp
0	0	1	001	1
0	1	0	010	2
0	1	1	011	3
1	0	0	100	4
1	0	1	101	5
1	1	0	110	6
1	1	1	111	7

Note: Please note that each unique `GROUPING` id only indicates the grouping set, which produced the groups in this grouping. It does not uniquely identify a group within this grouping!

Combining and Merging Grouping Sets

The group by clause may not only take simple group expressions or grouping sets, but also any combination of them, e. g., `GROUP BY a, ROLLUP(b,c),GROUPING SETS((d,e), (f))`. The resulting grouping sets are then defined by the Cartesian product of the involved grouping sets, i.e. the example clause translates into the following grouping sets:

This is the result of the Cartesian product of {(a)} x {(b, c), (b), ()} x {(d, e), (f)}

(a, b, c, d, e)

(a, b, c, f)

(a, b, d, e)

(a, b, f)

(a, d, e)

(a, f)

Likewise, the `GROUPING SETS` clause may take nested `ROLLUP`, `CUBE` and `GROUPING SET` clauses, e. g. `GROUP BY GROUPING SETS (a, ROLLUP(b,c), GROUPING SETS((d,e), (f)) )`. In this case, the nested grouping sets are "flattened" and merged into the surrounding grouping set, i.e., the example clause translates into the following grouping sets:

This is the result of the grouping sets (a), followed by all grouping sets (b,c), (b), () resulting from the rollup clause, followed by the grouping sets (d,e) and (f).

(a)

(b, c)

(b)

()

(d,e)

(f)

Having Clause to Filter Groups

The Having clause allows you to filter the rows in the query results based on aggregate caculations when the query includes a Group By clause. The example shown here returns rows for each grouped stock symbol only if the average open price is greater than 100:

```
<mashup name="testXMLNumbers"
 xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
 xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
 xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http
 <operation name="invoke">
  <output name="result" type="document"/>
  <variable name="stockType" type="schema">
  define dataset stocks (symbol string,
      open double,
      close double,
      adjclose double,
      low double,
      high double,
      volume double,
      datetime datetime 'EEE MMM dd hh:mm:ss Z yyyy')
  </variable>
  <variable datafile="stocks.xml" name="stocks" stream="true" type="variable:stockType
  <raql outputvariable="result">
  select symbol, avg(open) as open from stocks
  group by symbol having avg(open) &gt; 100.00
  </raql>
 </operation>
</mashup>
```

```
Run | Text view | Tree view | Consol
Performance Stats
<?xml version="1.0" encoding="UTF-8"?>
<records>
  <record>
    <symbol>FSLR</symbol>
    <open>143.91365957446806</open>
  </record>
  <record>
    <symbol>GOOG</symbol>
    <open>435.44892045454526</open>
  </record>
  <record>
    <symbol>IBM</symbol>
    <open>106.80763649425255</open>
  </record>
  <record>
    <symbol>ISRG</symbol>
    <open>118.00512323943663</open>
  </record>
</records>
```

You can use the same comparisons in Having clauses as you can in Where clauses.

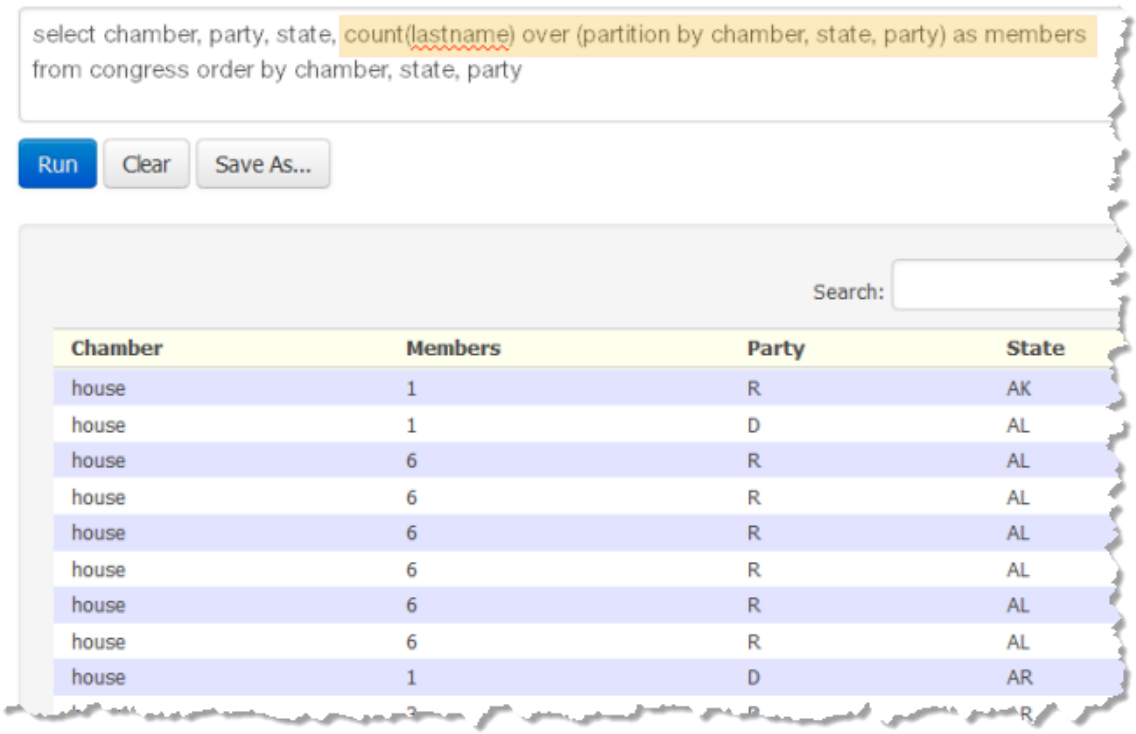
Partitions and Windows

The "Group and Analyze Rows with Row Detail" on page 701 example in Getting Started introduced a simple partition using the Over clause to perform some analysis and add these results to each row in the dataset. This topic includes examples of:

- Example 119. Multi-Level Partitions
- Example 120. Windows as Relative Subsets Within Partitions
- Example 121. Windows to Show Selected Sibling Values

Multi-Level Partitions

The column expression you use in a partition definition for Over clauses can use a list of multiple columns to define multi-level partitions, just like column expressions for Group By clauses. The following example segments US legislators into partitions for each chamber of Congress, state and political party to determine the number of legislators for each party in each state and legislative chamber:



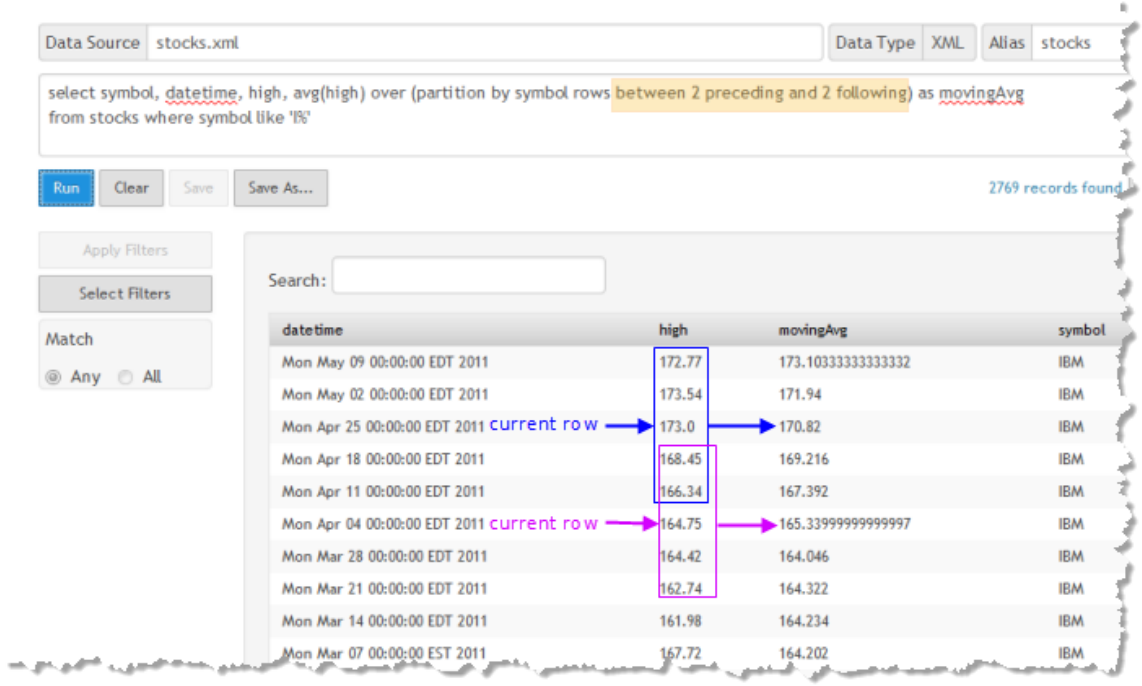
Windows as Relative Subsets Within Partitions

Windows define subsets of rows within a partition that are relative to the current row based on row position. You define windows by adding `rows between` to the partition definition, such as:

```
(partition by symbol rows between 2 preceding and 2 following)
```

This example is *centered* where the number of preceding and following rows is equal. Windows can be asymmetric, using different numbers of preceding and following rows or omitting either. Analytic functions are applied to just those rows within the window based on the current row and the result is added to the current row.

Windows are useful for time-based datasets, although they are not limited to this. With time-based datasets, each row represents a different slice of data for a specific time. The following example uses windows with a time-based dataset to calculate moving averages:



The moving average is calculated over the rows in the window relative to the current row so each row potentially has a different moving average. Moving averages typically show a smoother trend for the column.

Windows to Show Selected Sibling Values

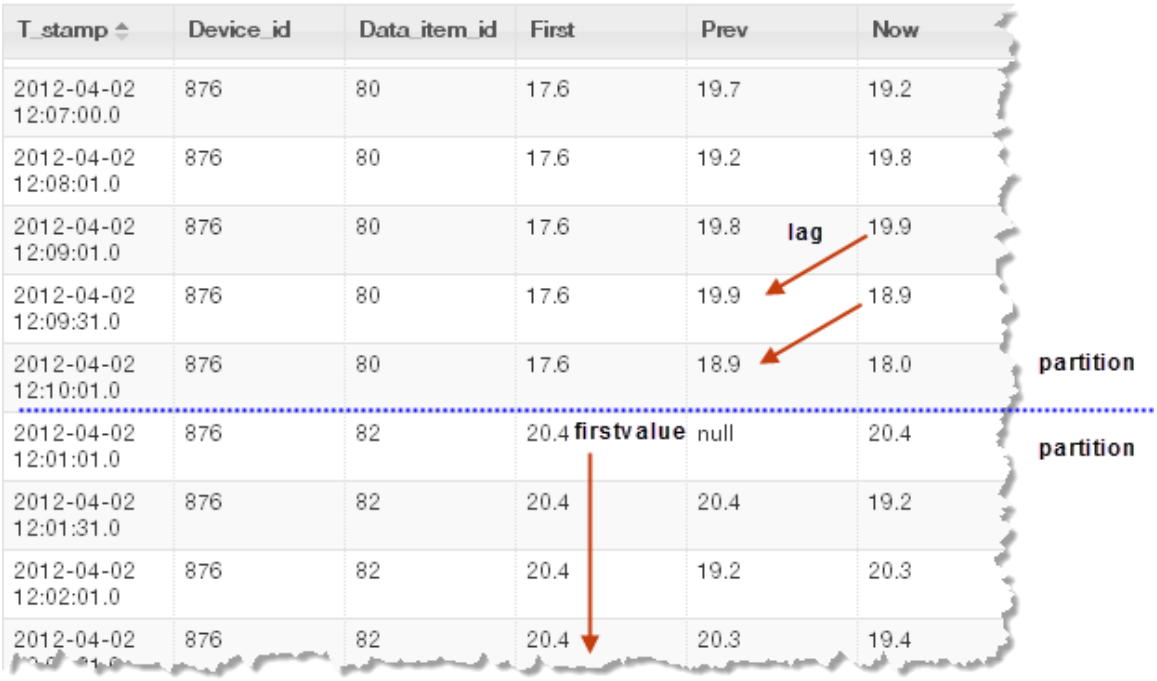
With window analytic functions, such as `lag()` or `firstvalue()`, you can select values for specific siblings for each row using window definitions. These functions return the column value for a sibling as shown in this example:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoSpec.xsd'
      xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
      xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
      xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
      name='lagfirstSensors'>

 <output name='result' type='document' />
 <variable name='readings' type='document' stream='true'/>

 <sql name='Sensors_Db' stream='true' outputvariable='readings'
     query="select device_id, data_item_id, data_value, t_stamp
          from sensors.readings_raw
          where (data_item_id = 80 or data_item_id = 82 or data_item_id=83) and
          t_stamp >= '2012-04-02 12:00:01' and t_stamp &lt; '2012-04-02 12:10:01' limit 100" />

 <raql outputvariable='result'
     query='select device_id, data_item_id, t_stamp, data_value as Now,
       lag(data_value) over (partition by data_item_id as rows between 10 preceding and 0 following) as Prev,
       first_value(data_value ) over (partition by data_item_id as rows between 10 preceding and 0 following) as First
       from readings'>
 </raql>
</mashup>
```

The `lag()` function selects the column value of the preceding sibling, while `first_value()` selects the column value of the first preceding sibling based on the

partition or window definition. In this example, Prev is added to each row, using `lag()` to get the last sensor reading for the current row. First is also added to each row and gets the value 5 seconds previous, based on the window size (10 preceding rows) and readings every half second.

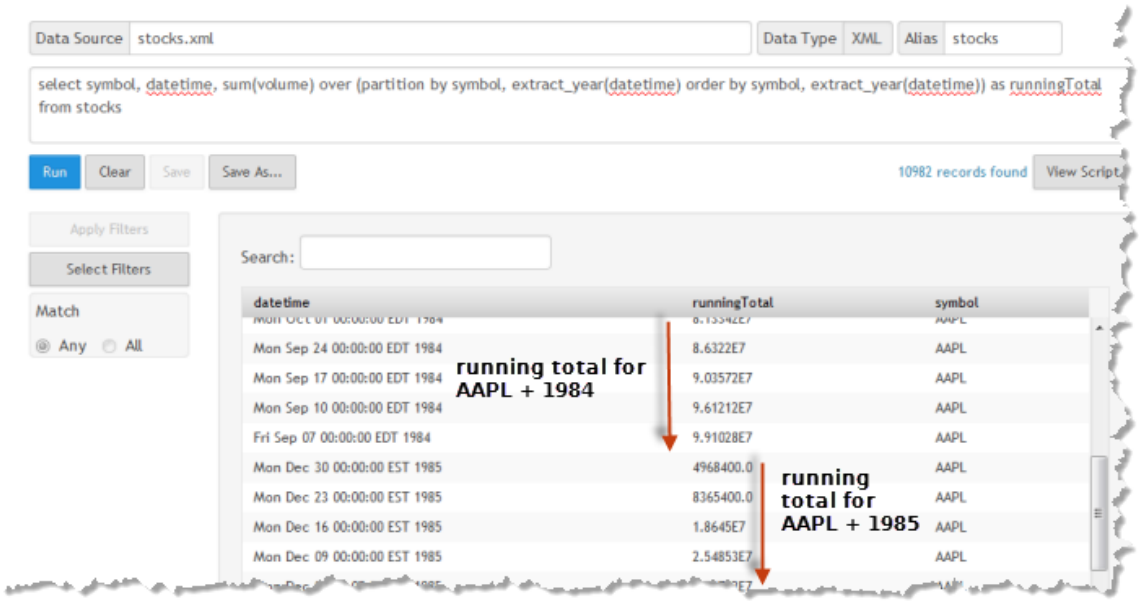
Here is an example of the results of this query:



Running Aggregates

You can get cumulative calculations, also known as running totals, using aggregate analytic functions over partitions or windows defined in Over clauses. Normally, aggregate analytic functions return a single value for all rows in a partition or window, such as the Example 119. Multi-Level Partitions example. If you add an Order By clause *within* the partition or window definition, however, aggregate functions return cumulative values based on the current row and all previous rows in the partition or window.

The following example uses the `sum()` aggregate analytic function to provide a running total of volumes for the stocks dataset introduced in Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics in Getting Started. The dataset is segmented into partitions by symbol plus the year in the date column and then ordered by the same column expression.



The runningTotal column is added to each row and calculated as a running sum for each symbol + year combination.

Number or Rank Rows

You can use any of these built-in MashZone NextGen analytic window functions to rank or number rows in partitions:

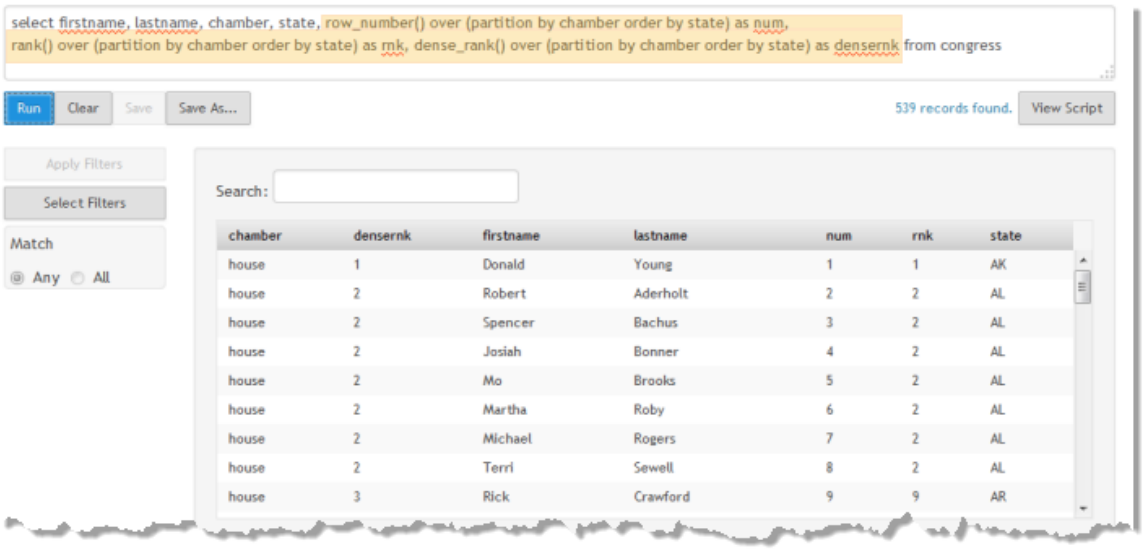
- `row_number()`
- `rank()`
- `dense_rank()`

To support ranking you must include an Order By clause *within* the partition definition. The first column in the internal Order By clause is used to determine row or ranking order.

You also *cannot* apply numering or ranking to windows within the partition.

Row_number, Rank and Dense_rank Example

This example uses all of these built-in ranking/numbering functions with the same partition (legislators within each chamber of the US congress) and sort order (ordered by state) to illustrate the different effects of each function:



- `row_number()` provides a simple sequential number for each row based on its position within the sorted partition. Note that this example is sorted, but that is not required for simple numbering.
- `rank()` assigns the same index number to all rows that match one unique value for the first sort column in the Order By expression. Rows are sorted within the partition based on the full Order By expression. The actual rank numbers are *not* contiguous, however, skipping unused numbers to reflect the number of rows within that rank.

  So in this example, the row number for the first legislator for Alabama is 2 and his rank is also 2. Rank for all the remaining legislators for Alabama remains at 2 although their row numbers continue to increment. When the first row for Arkansas is found, the row number increments to 9 and so does the rank, skipping ranks of 3-8.
- `dense_rank()` also assigns the same index number to all rows that match one unique value for the first sort column in the Order By expression. Unlike `rank()`, however, `dense_rank()` results *are* contiguous.

  So in this example, the rank number for the first legislator from Arkansas is 9, matching his row number. The dense rank number is 3, being the next available ranking number.

Analytic Functions for Partitions and Windows

Analytic functions are used in Select clauses to perform analysis for partitions or windows defined in Over clauses. You can use either aggregate analytic or window analytic functions. See "Built-In Analytic Functions: Aggregate and Window" on page 785 for a list of the MashZone NextGen built-in analytic functions available to you. User-defined analytic functions may also be available.

For examples of analytic functions used in partitions or windows see:

Aggregate Analytic Functions	Plain Analytic Functions
"Group and Analyze Rows with Row Detail" on page 701 for an example using the `correlation` statistical function.	"Example 122. Row_number, Rank and Dense_rank Example" on page 758 using all three of the built-in analytic functions that assign ranks or numbers to rows within a partition
"Example 120. Windows as Relative Subsets Within Partitions" on page 755 for an example calculating a moving average.	"Example 121. Windows to Show Selected Sibling Values " on page 756 to retrieve specific values relative to the current row.
"Running Aggregates" on page 757	

Joins and Other Multiple-Dataset Operations

You can work with multiple datasets in a RAQL query using any of the following operations:

- Join: Merge Columns for Dataset Rows to perform *inner joins* that combine columns from two or more datasets into the result for rows that match a specific condition. You can also perform natural joins, cross joins or outer joins, if needed.
- Union: Append Like Datasets to add the rows from two or more datasets into a larger result. The columns and datatypes of all datasets must be identical.
- Intersect: Find Common Dataset Members to find those rows in two or more datasets that are members of all the datasets. The columns and datatypes of all datasets must be identical.
- Except or Minus: Find All Less the Intersection to find those rows in the first dataset that are *not* members of the second dataset. The columns and datatypes of all datasets must be identical.

The basic query technique to work with multiple datasets uses a separate subquery for each dataset in the following form. This does not apply to join operations.

subquery-one `[ UNION | INTERSECT | EXCEPT | MINUS]` *subquery-two [... optional additional multi-set operators and additional subqueries ...]*

The individual subqueries for each dataset follow the same syntax and rules as usual, with one addition. You may need to include both the dataset and column names in the form *dataset-name.column-name* to clearly identify a column. (See "Valid Names for Datasets, Columns, Aliases, Paths and Functions" on page 727 for more information.)

With Join, this basic query technique is applied within the From clause instead, such as:

`SELECT` *columns-expression* `FROM` *subquery-one* `[ JOIN]` *subquery-two [... optional multi-set operator and subqueries ...]* `WHERE` *join-condition*

Join: Merge Columns for Dataset Rows

A Join clause works just like joins for relational databases. It finds rows in each dataset that match a specified condition and create result rows that join the columns from each dataset.

RAQL supports inner joins, natural joins, cross joins and outer joins. Inner joins are most common and thus are the default.

The Select clause identifies columns to include in the result from both datasets. Column names are prefixed by the dataset they belong to. The From clause identifies one dataset, followed by the Join clause that identifies the second dataset.

This example uses the `on` *condition* syntax to determine which rows to join, but RAQL also supports the `using` *column-list* syntax. If the column name for the condition is identical in both datasets, you could also use:

```
select employees.first_name, employees.last_name,
  employees.job_id, jobs.job_title
  from employees
  join jobs
  using job_id
```

RAQL supports also LEFT, RIGHT and FULL outer joins in standard SQL syntax.

Examples for Outer joins

LEFT OUTER JOIN

```
select *
FROM OpenAuction
LEFT OUTER JOIN ClosedAuction
ON OpenAuction.itemID<=ClosedAuction.itemID;
```

RIGHT OUTER JOIN

```
SELECT *
FROM OpenAuction
RIGHT OUTER JOIN ClosedAuction
ON OpenAuction.itemID<=ClosedAuction.itemID;
```

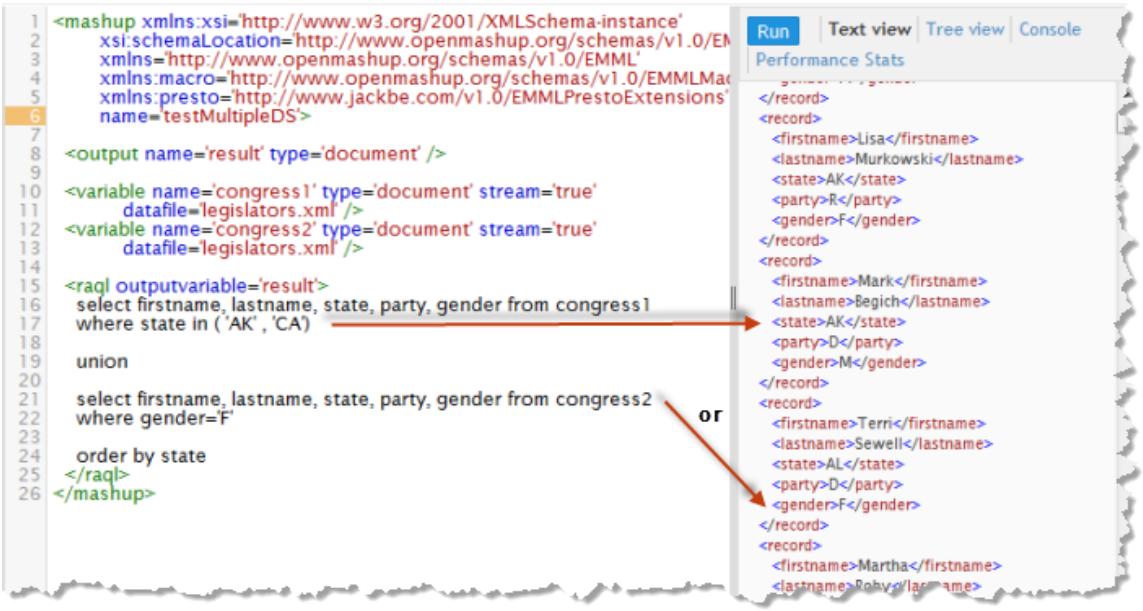
FULL OUTER JOIN

```
SELECT *
FROM OpenAuction
FULL OUTER JOIN ClosedAuction
ON OpenAuction.itemID<=ClosedAuction.itemID;
```

Union: Append Like Datasets

A Union clause appends rows from both subqueries into the result. The results of both subqueries must have identical columns and datatypes.

The following example merges a list of congress persons from Alaska or California with a list of female congress persons:




```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
        xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EM
        xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
        xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMac
        xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
        name='testMultipleDS'>

 <output name='result' type='document' />

 <variable name='congress1' type='document' stream='true'
           datafile='legislators.xml' />
 <variable name='congress2' type='document' stream='true'
           datafile='legislators.xml' />

 <raql outputvariable='result'>
   select firstname, lastname, state, party, gender from congress1
   where state in ('AK', 'CA')

   union

   select firstname, lastname, state, party, gender from congress2
   where gender='F'

   order by state
 </raql>
</mashup>
```

```
</record>
<record>
  <firstname>Lisa</firstname>
  <lastname>Murkowski</lastname>
  <state>AK</state>
  <party>R</party>
  <gender>F</gender>
</record>
<record>
  <firstname>Mark</firstname>
  <lastname>Begich</lastname>
  <state>AK</state>
  <party>D</party>
  <gender>M</gender>
</record>
<record>
  <firstname>Terri</firstname>
  <lastname>Sewell</lastname>
  <state>AL</state>
  <party>D</party>
  <gender>F</gender>
</record>
<record>
  <firstname>Martha</firstname>
```

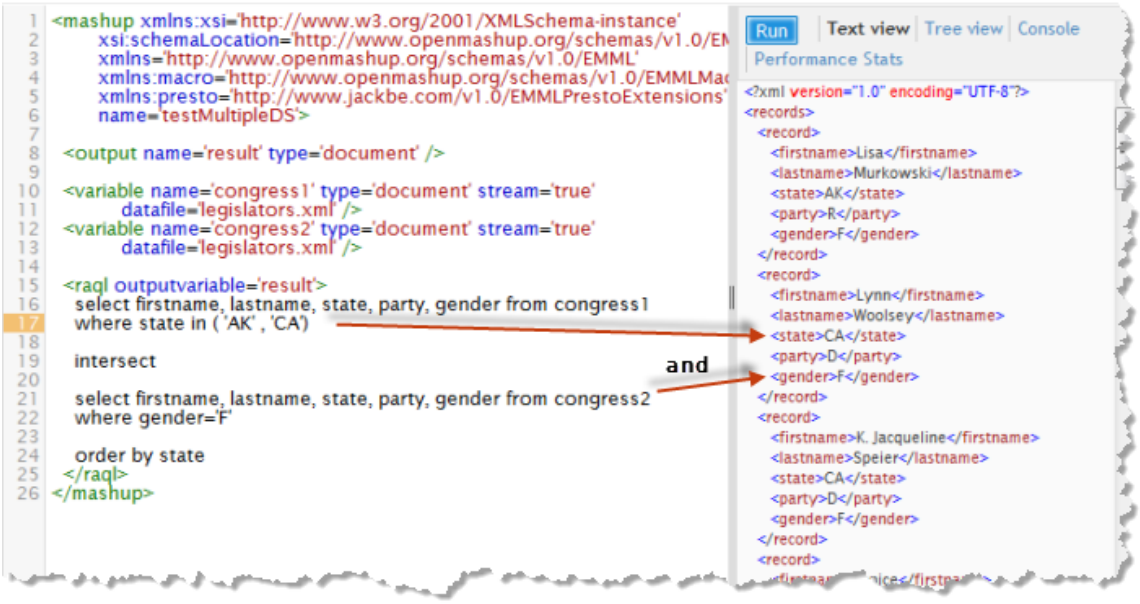
This example also illustrates an additional point. Duplicate rows are removed by default with Union (equivalent to using the Distinct keyword). So in this example, the female congress person from Alaska only appears once even though she is present in the results of both subqueries.

You can use the `ALL` keyword to keep duplicate rows if desired.

Intersect: Find Common Dataset Members

An Intersect clause finds rows that exist in the results of both subqueries. The results of both subqueries must have identical columns and datatypes.

The following example finds congress persons from Alaska and California that are also female:



Intersect also removes duplicate rows by default, but you can use the `ALL` keyword to keep duplicate rows if desired.

Except or Minus: Find All Less the Intersection

An Except clause, or the alternate Minus operator, selects all members of the first subquery that are *not* also in the second subquery. The results of both subqueries must have identical columns and datatypes.

The following example finds congress persons from Alaska and California that are not also female:

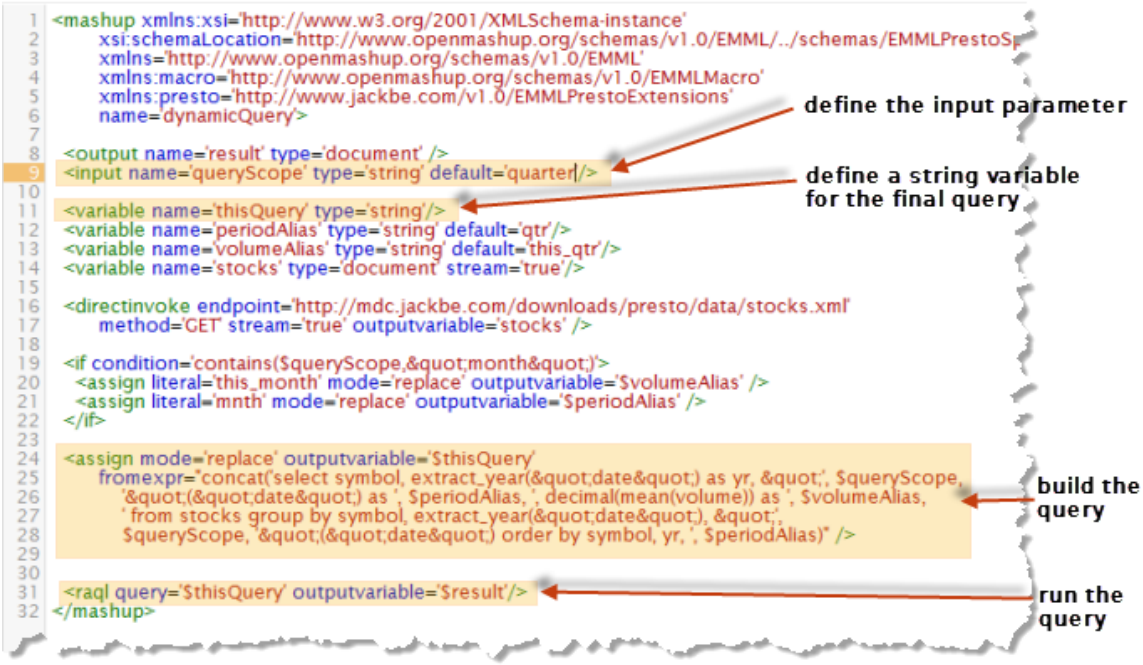
Except also removes duplicate rows by default, but you can use the `ALL` keyword to keep duplicate rows if desired.

Creating Dynamic RAQL Queries

At its simplest, dynamic queries use input parameters to provide the values used in Where clause conditions using dynamic mashup expressions. See "Parameters in Where Clauses" on page 739 for an example.

But sometimes, you need more flexibility to make other clauses be dynamic. For these more demanding cases, you can build entire RAQL queries using the EMML<assign> statement, the `concat()` XPath function, and other EMML statements.

The following example builds the entire RAQL query based on an input parameter that chooses the time period to use for grouping stock volumes from the example dataset introduced in Getting Started.




```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLPrestoS
    xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
    xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
    xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
    name='dynamicQuery'>

 <output name='result' type='document' />
 <input name='queryScope' type='string' default='quarter'/>

 <variable name='thisQuery' type='string'/>
 <variable name='periodAlias' type='string' default='qtr'/>
 <variable name='volumeAlias' type='string' default='this_qtr'/>
 <variable name='stocks' type='document' stream='true'/>

 <directinvoke endpoint='http://mdc.jackbe.com/downloads/presto/data/stocks.xml'
    method='GET' stream='true' outputvariable='stocks' />

 <if condition='contains($queryScope,&quot;month&quot;)'>
   <assign literal='this_month' mode='replace' outputvariable='$volumeAlias' />
   <assign literal='mnth' mode='replace' outputvariable='$periodAlias' />
 </if>

 <assign mode='replace' outputvariable='$thisQuery'
    fromexpr="concat('select symbol, extract_year(&quot;date&quot;) as yr, &quot;', $queryScope,
      '&quot;(&quot;date&quot;) as ', $periodAlias, ', decimal(mean(volume)) as ', $volumeAlias,
      ' from stocks group by symbol, extract_year(&quot;date&quot;), &quot;',
      $queryScope, '&quot;(&quot;date&quot;) order by symbol, yr, ', $periodAlias)" />

 <raql query='$thisQuery' outputvariable='$result'/>
</mashup>
```

This example uses that parameter to determine both the functions and aliases for fields to use in the Select clause as well as functions used in the Group By and Order By clauses. To do this, the mashup:

- Defines am input parameter, `queryScope`, with the `<input>` statement to get the time period of interest. In this case it can be either `quarter` or `month`.

  You can also use other EMML statements to provide the dynamic values needed.

- Defines a variable, `thisQuery`, in a `<variable>` statement with a string datatype to hold the full query string.

You can also define the literal portions of the query in other string variables by assigning literal content as their default value.

- Defines string variables for the alias names to be used in the query. Both `volumeAlias` and `periodAlias` have default values that correspond with the default value for `queryScope`.

  These variables are altered to appropriate month names in the `<if>` statement when the `queryScope` parameter is set to month.

- Combines the literal portions of the query with the dynamic portions using the `<assign>` statement and the `concat()` XPath function.

  For the literal portions of the query, you can supply this directly as a string in the arguments to `concat()`, as this example does. Or you can refer to other variables that contain this text.

  You refer to variables in the function, or in other EMML statements, using `$`*variable-name* .

- Refers to the variable with the completed query, using `$`*query-variable-name* , in the `query` attribute of the <raql> statement.

Depending on the value of the `queryScope` parameter, the actual RAQL query that this mashup uses is either:

```
select symbol, extract_year(datetime) as yr,
quarter(datetime) as qtr, CAST(avg(volume) AS DOUBLE) as this_qtr
from stocks
group by symbol, extract_year(datetime), quarter(datetime)
order by symbol, yr, this_qtr
```

or

```
select symbol, extract_year(datetime) as yr,
"month"(datetime) as mnth, CAST(avg(volume) AS DOUBLE) as this_month
from stocks
group by symbol, extract_year(datetime), "month"(datetime)
order by symbol, yr, mnth
```

Migrate RAQL Queries from Version 3.7 to 3.8

With this release, there are several significant changes to RAQL that require changes to existing RAQL queries from 3.7 or earlier. Edit mashups to:

- Update <storeto> Syntax to store datasets in separate, named In-Memory Stores. The original syntax which store multiple datasets in the same store is *no longer* supported.
- Add Schemas or Update Datatypes for CSV or XML datasets. Changes in implicit casting may cause existing queries to break without this additional meta-data.
- Update Built-In Plain Functions for 3.8 for queries that use deprecated functions or functions that are *no longer* supported.

> **Note:** In a few cases, the functions listed are supported but the parameters or the semantics of the function have changed sufficiently that queries may need updates to perform as expected.

- Update Built-In Aggregate or Window Analytic Functions for 3.8 for queries that use deprecated functions or functions that are *no longer* supported.

> **Note:** In a few cases, the functions listed are supported but the parameters or the semantics of the function have changed sufficiently that queries may need updates to perform as expected.

- Implement and Replace User-Defined Functions based on the new RAQL User Defined Functions API. This API has changed completely, so user-defined functions from previous versions must be updated.

Update <storeto> Syntax

The original syntax for storing datasets in an In-Memory Store allowed you to store multiple datasets in the same store with each dataset as one entry. This syntax and behavior is *no longer* supported in 3.8.

You must update any RAQL query that uses this syntax to store the dataset in a separate, named In-Memory Store. See "<storeto>" on page 882 for details on the new syntax and links to examples.

In addition, the underlying mechanism used for storing datasets in an In-Memory Store has been extended. Therefore, search attributes must not be configured explicitly. Instead, the cache must be configured as being searchable and dynamic indexing must be enabled (use `<searchable allowDynamicIndexing="true" />` in cache configuration file) and the search attributes must be specified in the "<storeto>" on page 882 statement using the searchattributes attribute.

Add Schemas or Update Datatypes

With version 3.8, RAQL has added, and in some cases, changed functionality to more closely follow SQL. As a result, queries on datasets that do not have datatype and schema information may break as some of the implicit casting from previous releases is no longer supported.

With CSV or XML datasets, you may need to provide datatype and schema information to permit existing queries to work correctly. You can add this meta-data to mashups directly, or add them as MashZone NextGen global attributes that you can refer to in many mashups. See "Providing Dataset Path and Datatype Information in a Schema" on page 722 for instructions.

In addition, the `decimal` datatype from 3.7 and earlier is *no longer* valid. Update any existing schemas with `double` or `bigdecimal` instead. See "Valid RAQL Datatypes" on page 772 for a complete list of supported datatypes.

Update Built-In Plain Functions for 3.8

Deprecated or Unsupported 3.7 Functions	Replace with 3.8 Function
`concat(String column-or-literal, String column-or-literal)`	Use the `||` operator
`"date"(String column)` `"date"(String column, String format)`	`to_date`

`"day"(Date column)`
`"day"(String column)`
`"day"(Long column)`
	`extract_day`

`decimal(Double column)` **(Unsupported Function)**
`decimal(Long column)`
`decimal(String column)`
	`cast(`*`column`*`as `*`datatype`*`)`

`"hour"(Date column)`
`"hour"(String column)`
`"hour"(Long column)`
	`extract_hour`

`length(String column)` **(Unsupported Function)**	`char_length` or `character_length`

`"minute"(Date column)`
`"minute"(String column)`
`"minute"(Long column)`
	`extract_minute`

`"month"(Date column)`
`"month"(String column)`
`"month"(Long column)`
`"month"(Object column)`
	`extract_month`

`number(String column)` **(Unsupported Function)**	`cast(`*`column`*`as `*`datatype`*`)`

Deprecated or Unsupported 3.7 Functions	Replace with 3.8 Function

`round(Object column)`	This function no longer accepts objects and implicitly casts them to numeric values.
	Mashups that relied on this implicit casting must update queries to use the following:
	`round(CAST(Object column AS double))`
	If no number of decimal places are specified, this function now rounds to zero decimals which may also affect mashup results. Mashups that require two decimal places must update queries to use the following:
	`round(Number column, "2")`

`"second"(Date column)`
`"second"(String column)`
`"second"(Long column)`
	`extract_second`

`"string"(String column)`	`cast(`*`column`*`as `*`datatype`*`)`
`substr(String column, Integer begin, Integer end)`	`substring(String column, Integer start-position, Integer length` **Note:** This is *not* a simple substitution as the semantics of the parameters that identify the characters to extract have different meanings in these functions. In addition to different parameters, character positions in `substr` use zero-based indexes while

Deprecated or Unsupported 3.7 Functions	Replace with 3.8 Function
	`substring` uses 1-based indexes.
`to_long(String column)`	`cast(`*`column`*`as` *`datatype`*`)`

`"week"(Date column)`
`"week"(String column)`
`"week"(Long column)`
	`extract_week`

`"year"(Date column)`
`"year"(String column)`
`"year"(Long column)`
`"year"(Object column)`
	`extract_year`

For more information on any built-in plain function, see "Built-In RAQL Functions" on page 775

Update Built-In Aggregate or Window Analytic Functions for 3.8

Deprecated or Unsupported 3.7 Functions	Replace with 3.8 Function
`correlation(Number column, Number column)`	`corr(Number column, Number column)`
`covariance(Number column, Number column)`	`covar(Number column, Number column)`
`denserank()`	`dense_rank()`
`firstvalue(Object column)`	`first_value(Object column)`
`analytics.kmean_clusters(String column-list, Integer k, Integer iterations, String measure)`	`analytics.kmeans_clusters(String column1[,String column2,...String columnN]; Integer k, Integer iterations, String measure)` This changes the function name (for consistency) and

Deprecated or Unsupported 3.7 Functions	Replace with 3.8 Function
	changes the signature of the function. The syntax change supports a variable number of columns as parameters to identify the features for clustering.
`analytics.kmeans_observations(String column-list, Integer k, Integer iterations, String measure)`	`analytics.kmeans_observations(String column1[,String column2,...String columnN]; Integer k, Integer iterations, String measure)` This changes the signature of the function to support a variable number of columns as parameters to identify the features for clustering.
`lastvalue(Object column)`	`last_value(Object column)`
`mean(Number column)` **(Unsupported Function)**	`avg(Number column)`
`rownumber()`	`row_number()`

For more information on any built-in analytic function, see "Built-In RAQL Functions" on page 775

Implement and Replace User-Defined Functions

The RAQL User-Defined Function API has changed completely for version 3.8. See the MashZone NextGen RAQL User-Defined Function API for reference information on this API in version 3.8.

The previous API is *no longer* supported. If you have implemented and deployed user-defined functions for RAQL in version 3.7 or earlier, you must re-implement these functions using the new API and redeploy them in version 3.8. See "Create and Add User-Defined Functions for RAQL Queries" on page 804 for instructions and examples of functions based on this new API.

RAQL Datatypes and Data Formats

Supported Data Formats for RAQL

RAQL can perform queries on datasets in the following formats:

- *CSV* (comma-separated values):
  - The first line in a CSV dataset must identify column names for the data.
  - RAQL replaces any white space character in column names with an underscore (_). For example, "First Name" becomes First_Name.
  - Column names that are numeric have "column_" prepended to the name. For example, "2010" becomes "column_2010".
- **JDBC Result Sets**: returned from databases when SQL queries or stored procedures are invoked.
- **XML**: data must be well-formed. In addition, RAQL has the following limitations for XML data:
  - XML namespaces are ignored.
  - The structure of the XML should be flat, with a single set of repeating nodes (the rows) that contain a single level of elements (the columns) with simple content (text only). Data in any nodes that are ancestors of the repeating 'rows' is not accessible.

    For example:

    ```
    <records>
      <record>
        <itemId>N2390</itemId>
        <price>145.20</price>
        ...
      </record>
      <record>
        <itemId>G88</itemId>
        <price>16.95</price>
        ...
      </record>
      ...
    </records>
    ```

  - Data in attributes may not be accessible in some situations.
- **JSON**: data must be well-formed. The structure of the JSON should be flat, with a single array of objects (the rows) that contain name/value pairs (the columns) with simple content (number, string, boolean). Data in any objects that are ancestors of the repeating 'rows' is not accessible.

  For example:

  ```
  {
      "records": {
          "record": [
  ```

```
            {
                "itemId": "N2390",
                "price": 145.2,
                ...
            },
            {
                "itemId": "G88",
                "price": 16.95,
                ...
            },
            ...
        ]
    }
}
```

Assuming that the above JSON data is available in file sales.json, the following EMML sample executes RAQL on it:

```
<mashup xmlns='http://www.openmashup.org/schemas/v1.0/EMML'>
  <variable name='sales'
    datafile='sales.json'
    type='document'
    subtype='json'
    stream='true'/>
  <output name='result'
    type='document' />
  <raql outputvariable='result'
    query='select itemId, price from sales/records/record'/>
</mashup>
```

- **Java Objects**: loaded in In-Memory Stores by external systems. Java objects must:
  - Be plain Java objects or beans with properties for each column of data in the dataset.
  - Be serializable. This is required when In-Memory Stores use both local memory for the MashZone NextGen Server and memory from additional BigMemory hosts. See "In-Memory Dataset Management" on page 861 for more information.
  - Have search attributes defined in the configuration for the declared In-Memory Store where they will be stored. Search attributes provide the extraction class and other information that maps Java object properties to dataset columns and allows RAQL to access and work with the data.

Valid RAQL Datatypes

RAQL supports the following simple datatypes for column data:

RAQL Datatype	Java Equivalent
bigdecimal	`java.math.BigDecimal`
boolean	`java.lang.boolean`
byte	`java.lang.byte`

RAQL Datatype	Java Equivalent
character	`java.lang.character`
datetime	`java.util.Date`
double	`java.lang.Double`
float	`java.lang.Float`
integer	`java.lang.Integer`
long	`java.lang.Long`
short	`java.lang.Short`
string	`java.lang.String` Data from CSV or XML datasets is generally treated as String. You can, however, add datatype meta data by "Providing Dataset Path and Datatype Information in a Schema" on page 722.

JDBC results sets can contain columns with complex data, such as BLOB, CLOB or Struct and these columns can be included in the query result. However, the query itself cannot access complex data to perform comparisons or calculations.

RAQL Operators

Arithmetic Operators

You may use the standard arithmetic operators in RAQL expressions: `+` , `-` , `*` or `/` .

Note: The division operator uses integer division *unless* the datatype of the data is decimal. You can set the datatype using casting functions or by additing datatype information in a schema. See "Built-In Plain Functions" on page 775 and "Providing Dataset Path and Datatype Information in a Schema" on page 722 for more information.

Comparison Operators

You may use the following comparison operators in Where clauses:

- = or !=
- < or <=

- > or >=
- `like`: for matching text based on a pattern. Use `%` to indicate zero to any number of characters. For example: `where lastname like 'A%'` to find any last name starting with a capital A.
- `in`: to match values against an enumerated set of values defined within brackets. For example: `where direction in ("N","NW","W")`

The literal values or other expressions being compared *must* be the same basic datatype. In many cases, this may require that values or expressions be cast to an appropriate type for a successful comparison. See "Literal Values in Conditions or Expressions" on page 739 for additional information.

Text comparison with the mathematic comparison operators is not case sensitive. Text comparisons to patterns or enumerations are case sensitive.

Logical Operators

You may use the following logical operators in Where clauses:

- `and`
- `or`
- `not`

Operators are not case sensitive. You can use parentheses to build complex logical expressions such as: `where (service_level in ("gold","silver")) and (rating >= 2.0 or overdue < 100 ).`

String Operators

You may use the `||` string operator to concatenate strings or the `concat` function.

Case Operators

RAQL supports two forms of case operators.

The first form selects a result expression based on different values of a case expression. Its syntax is

```
case expression when expression then expression [,
                when expression then expression ]...
               [else expression]
                end
```

Example:

```
case status when 0 then 'success'
            when 1 then 'fail'
            else 'unknown'
end
```

The second form selects a result expression based on a boolean expression. Its syntax is

```
case when booleanExpression then expression [,
     when booleanExpression then expression ]...
    [else expression]
```

```
    end
```

Example:

The following example is equivalent to the above example, but uses the second form.

```
case when status=0 then 'success'
     when status=1 then 'fail'
     else 'unknown'
end
```

Built-In RAQL Functions

MashZone NextGen provides both Built-In Plain Functions and Built-In Analytic Functions: Aggregate and Window that you may use in RAQL queries.

Note: Some built-in functions from version 3.7 are deprecated or no longer supported in version 3.8. See "Update Built-In Plain Functions for 3.8" on page 767 for a complete list and the alternate functions from version 3.8 to use as replacements.

You can also define and add your own plain or analytic functions to RAQL. See "Create and Add User-Defined Functions for RAQL Queries" on page 804 for instructions.

Built-In Plain Functions

Plain functions can be used in Select, Over, Where, Order By, Group By and Having clauses in RAQL queries. They typically either *cast* (change) the datatype of column values, extract part of the values or transform values in some way.

Plain functions are applied individually to each value in the column specified *without* access to values in other rows.

Function	Description
`abs(Number col-or-expr)`	Returns the absolute value of the numeric value for the column or expression.
`cast(Any col-or-expr AS datatype)`	Casts the values of the specified column or expression to the specified datatype. See "Valid RAQL Datatypes" on page 772 for a list of valid datatypes.
`ceiling (Number col-or-expr)`	Rounds the numeric value for the column or expression up to the next highest integer value.
`char_length (String col-or-expr)`	Returns the length of the values in this column or expression as an integer. Returns `null` if the column value is null.

Function	Description
`character_length (String col-or-expr)`	**Note:** in 3.7 and earlier releases, this returned `0` if the column value was null.
`concat(String column-or-literal, String column-or-literal)`	*Deprecated*. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives. If the value of any column or expression is null, this returns a null value.
`custom_month(String col-or-expr)`	Extracts the month from dates in the format *EEE MMM dd HH:mm:ss z yyyy* and returns the number of the month as an integer. This is based on English month names.
`"date"(String col-or-expr)` `"date"(String col-or-expr, String format)`	*Deprecated*. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.
`date_to_epoc_sec`	Represents internally the Java notation of time as milliseconds since midnight, January 1, 1970 UTC. ■ Conversion of date to seconds (not milliseconds!) since above reference time ■ Resulting double format compatible with Apama date storage in fractional seconds Examples ■ `date_to_epoc_sec(1970-01-01'T'13:33:11.000)`

`"day"(Date col-or-expr)`
`"day"(String col-or-expr)`
`"day"(Long col-or-expr)`
	Deprecated. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

`dayOfWeek(Date col-or-expr)` `dayOfWeek(String col-or-expr)`	Converts the values for this column or expression to a date and returns the day of the week as an integer [1 - 7]. This number is specific to the locale for the MashZone

Function	Description
`dayOfWeek(Long col-or-expr)`	NextGen Server. For example, day 1 is Sunday in the United States but Monday in France.
`epoc_sec_to_date`	Represents internally the Java notation of time as milliseconds since midnight, January 1, 1970 UTC. ■ Conversion of seconds (not milliseconds!) since above reference time to date ■ Three variants of function using either string, double, or long as input Examples ■ `epoc_sec_to_date('123456')` ■ `epoc_sec_to_date(123456.00)` ■ `epoc_sec_to_date(123456)`
`exp(Number col-or-expr)`	Raises `e`, the natural log base, to the power specified by the numeric value or expression passed.
`extract_date(Date dt, String field)`	Extracts the portion of the date or time from values for this column or expression that is identified by the *dt* parameter. Valid field values are: ■ `day` ■ `day-of-week` ■ `hour` ■ `minute` ■ `month` ■ `monthname` ■ `quarter` ■ `second` ■ `week` ■ `year`

Function	Description

`extract_hour(Date dt)`
`extract_hour(String dt)`
`extract_hour(Long dt)`
	Extracts the hour portion of the date and time in the column or expression that is identified by the *dt* parameter.

`extract_minute(Date dt)`
`extract_minute(String dt)`
`extract_minute(Long dt)`
	Extracts the minute portion of the date and time in the column or expression that is identified by the *dt* parameter.

`extract_month(Date dt)`
`extract_month(String dt)`
`extract_month(Long dt)`
	Extracts the month portion of the date and time in the column or expression that is identified by the *dt* parameter.

`extract_second(Date dt)`
`extract_second(String dt)`
`extract_second(Long dt)`
	Extracts the second portion of the date and time in the column or expression that is identified by the *dt* parameter.

`extract_week(Date dt)`
`extract_week(String dt)`
`extract_week(Long dt)`
	Extracts the week portion of the date and time in the column or expression that is identified by the *dt* parameter.

`extract_year(Date dt)`
`extract_year(String dt)`
`extract_year(Long dt)`
	Extracts the year portion of the date and time in the column or expression that is identified by the *dt* parameter.

`floor (Number column)`	Rounds the numeric value or expression down to the next closest integer value.
`format_date(dateColumn, datePattern, TimeZone)`	Represents string of date values in target format and target timezone. ■ datePattern may be 'short', 'medium', 'long', 'full' or any date pattern as specified in Java's SimpleDateFormat. ■ TimeZone may be any time zone as specified in Java's SimpleDateFormat

Function	Description
	Examples ■ `format_date ( dateColumn , ' yyyy - MM- dd ', 'GMT')` ■ `format_date ( dateColumn , ' h:mm a ', 'PST-08:00')`
`format_date (dateColumn, datePattern)`	Corresponds to `format_date(dateColum, datePattern, 'GMT')`
`format_date (dateColumn)`	Corresponds to `format_date(dateColum, 'yyyy-MM-dd'T'HH:mm:ss.SSSZ', 'GMT')`
`geo_distance(String column-or-lat1, String column-or-long1, String column-or-lat2, String column-or-long2)`	Calculates the geographical distance between two locations identified by latitude and longitude. Commonly, one set of coordinates is provided from two columns or expressions in the data and the other is provided as literal values. If any parameter value is null, this returns a null value.

`"hour"(Date col-or-expr)`
`"hour"(String col-or-expr)`
`"hour"(Long col-or-expr)`
	Deprecated. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

`lower(String col-or-expr)`	Converts the values for this column or expression to all lower-case characters.
`ln(Number col-or-expr)`	Calculates the natural log of the numeric value or expression. Column values or expressions *must* be greater than zero.
`matches(String col-or-expr, String pattern)`	Determines if a string matching the regular expression in *pattern* exists *anywhere* within the values of the column or expression. Matching is case insensitive. Returns a boolean value (`true` or `false`). For example:

Function	Description
	`where matches(firstname,'ra[n|l]')='true'` Will match Frank, Gerald or Randy in the firstname column. To perform exact regular expression matches, with no implicit wildcards, see the `regex` function.

`"minute"(Date col-or-expr)`
 	
`"minute"(String col-or-expr)`
 	
`"minute"(Long col-or-expr)`
	Deprecated. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

`mod(Number dividend-col-or-expr, Number divisor-col-or-expr)`	Returns the remainder of dividing the numeric value or expression of the dividend by the numeric value or expression of the divisor.

`"month"(Date col-or-expr)`
 	
`"month"(String col-or-expr)`
 	
`"month"(Long col-or-expr)`
 	
`"month"(Object col-or-expr)`
	Deprecated. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

`nvl(String col-or-expr, String str)`
 	
`nvl(Date col-or-expr, String str)`
 	
`nvl(Number col-or-expr, String str)`
	Replaces null values in the specified column or expression with the specified string value.
	
	Note: In previous releases, string columns with only white space characters (space, tab, etc.) were treated as a null value.
	
	Effective in 3.8, columns with string values are considered to be null only if they have no value or they contain an empty string.

`power(Number base-col-or-expr, Number exponent-col-or-expr)`	Calculates the value of the base number or expression raised to the power of the exponent value or expression.

Function	Description

`quarter(Date col-or-expr)`
`quarter(String col-or-expr)`
`quarter(Long col-or-expr)`
	Converts the values for this column or expression to a date and returns the number of the quarter as an integer.

`regex(String col-or-expr, String pattern)`	Determines if a string matching the regular expression in *pattern* exists in the values of the column or expression as specified. Matching is case insensitive. Returns a boolean value (`true` or `false`). **Note:** This function does not add any implicit wildcards to the regular expression. To find a string *anywhere* within column values, use the `matches` function. For example: `where regex(firstname,'ra[n|l]')='true'` Will match Ralph or Randy in the firstname column, but will not match Frank or Gerald.
`round(Number val)` `round(Number val, Integer decimal-places)`	Rounds the numeric value or expression to 0 decimal places, if no decimal places are specified, or to the number of decimal places passed as an argument. If any value is null, this returns a null value. **Note:** This function has changed in 3.8 in two respects: ■ In 3.7 and earlier releases, this function rounded the results to 2 decimal places if no decimal places were specified. ■ Column or expression values *must* be numeric. This function no longer implicitly casts values.
`"second"(Date col-or-expr)`	*Deprecated*. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

Function	Description
`"second"(String col-or-expr)` `"second"(Long col-or-expr)`	
`split_part(String col-or-expr, String delim, Integer part)` `split_part(String col-or-expr, Character delim, Integer part)`	Returns the *n* th part of the values of this column or expression when each value is split into parts at each delimiter defined in *delim* . For example: `split_part("INV:2012:GHI345",":",3)` Returns the string "GHI345".
`split_regex((String col-or-expr, String pattern, Integer part)`	This function is identical to `split_part` with the exception that the delimiter used to split the column values from *pattern* is a regular expression.
`sqrt(Numer col-or-expr)`	Calculates the square root of the specified numeric value or expression.
`"string"(String col-or-expr)`	*Deprecated*. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.
`substr(String col-or-expr, Integer begin, Integer end)`	*Deprecated*. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.
`substring(String col-or-expr FROM Integer start-position [FOR Integer length])`	Extracts the number of characters specified as *length* from the value or expression starting from the *start-position* . If *length* is not specified, extracts all characters from the *start-position* to the end of the string. Character positions are one-based (the first character is 1).
`time_mask(Date column, String mask)` `time_mask(String column, String mask)`	Converts the value for this column or expression to a date and time, if needed, in the form *yyyy-MM-dd:HH:mm:ss* .

Function	Description
	Note: If the values for the column do not contain times, the time is set to `00:00:00`. It extracts the portion of this date and time through the time part identified by the *`mask`*. Valid masks include: ■ `y` = returns only the year. ■ `M` = returns the year and month. ■ `d` = returns the year, month and day. ■ `H` = returns the full date and hour. ■ `m` = returns the full date, hour and minute. ■ `s` = returns the full date and time.

`to_date(String col-or-expr)`
 	
`to_date(String col-or-expr, String format)`
	Casts the values for this column or expression to `Date`.
	
	If no format pattern is provided, this function can have a negative impact on performance as it attempts to convert the string iterating over the following date formats:
	■ `M/dd/yy` or the short format for the current locale
	■ `MMM dd, yyyy` or the long format for the current locale
	■ `EEE MMM dd, yyyy` or the full format for the current locale
	■ `yyyy-MM-dd`
	■ `EEE MM dd HH:mm:ss z yyyy`
	■ `EEE, dd MMM yyyy HH:mm:ss Z`
	■ `yyyy-MM-DD'T'HH:mm:ss'Z'`
	
	Use the *`format`* parameter to define the date and time pattern used in values for this column.
	
	Note: For more information on date format patterns, see the Java `SimpleDateFormat` class.

Function	Description
`to_long(String col-or-expr)`	*Deprecated*. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.
`trim( [LEADING | TRAILING | BOTH] [String character] [FROM] String target-col-or-expr)`	Removes the specified character from the string values of the target column or expression in the specified locations: ■ `LEADING` = removes any number of the specified character at the beginning of the target string. ■ `TRAILING` ■ `BOTH` = removes any number of both leading and trailing characters from the target string. If no location is specified, this is the default. If the character to remove is not specified, this defaults to removing space characters.
`truncate(Number col-or-expr, Number decimals)`	For numeric values or expressions, this truncates the number to the number of decimals places specified. If the decimals parameter is omitted, truncates the number to zero decimals.
`upper(String col-or-expr)`	Converts the values for this column or expression to all upper-case characters.

`"week"(Date col-or-expr)`
`"week"(String col-or-expr)`
`"week"(Long col-or-expr)`
	Deprecated. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

`week_of_month(Date col-or-expr)`
`week_of_month(String col-or-expr)`
`week_of_month(Long col-or-expr)`
	Converts the values for this column or expression to a date and returns the number of the week for this month as an integer.

Function	Description

`"year"(Date col-or-expr)`
`"year"(String col-or-expr)`
`"year"(Long col-or-expr)`
	Deprecated. See "Update Built-In Plain Functions for 3.8" on page 767 for alternatives.

Built-In Analytic Functions: Aggregate and Window

Analytic functions most commonly perform calculations using sets of rows within a dataset. This may be the entire dataset or specific sets of rows defined as *groups*, *partitions* or *windows*.

There are two types of analytic functions:

- **Aggregate analytic functions**: use all rows in the dataset, group, partition or window to perform a calculation and return a single value. For example, `sum` adds the values of all rows in the current scope.

  Aggregate analytic functions can be be used in the Select or Having clauses of RAQL queries:

  - To return a single value for each group defined in a Group By clause.
  - To return a single value for each partition or each window defined in an Over clause. This single value is added to each row in the partition or window. You can also use aggregate analytic functions to return running calculations for partitions or windows defined in an Over clause.
  - To return a single value for the entire dataset if no group definition or partition definition is specified.

- **Window analytic functions**: use some of the rows in a partition or window to perform a calculation and return a value for each row. Typically, this uses rows that are relative to the current row, such as `first_value` or `row_number`.

  You must include an Over clause in queries that use window analytic functions. This defines the partitions or windows used by the function.

Aggregate	Window	Function	Description
✓		`avg(Number col-or-expr)`	Returns the average value from the specified column or expression for all rows within the group, partition or window scope. This ignores any null values.

Aggregate	Window	Function	Description
✓		`correlation(Number col-or-expr, Number col-or-expr)`	*Deprecated*. See "Update Built-In Aggregate or Window Analytic Functions for 3.8" on page 769 for alternatives.
✓		`corr(Number col-or-expr, Number col-or-expr)`	Returns a correlation coefficient (Pearson's product-moment) for the values in the two columns or expressions for this partition or for the current window in this partition. Null values are ignored. This coefficient indicates whether the two columns have a linear relationship. The closer the value is to `|1|` the closer the relationship is to linear. If the value returned is: ■ `0`: no linear relationship exists. ■ `0 < x <=1`: a linear relationship exists where both columns increase as the other increases. ■ `-1 <= x < 0`: there is an inverse linear relationship. One column decreases as the other increases.
✓		`count(Object col-or-expr)`	Returns the number of rows for the specified column or expression for all rows within the group, partition or window scope. Rows with null values are ignored.
✓		`covariance(Number col-or-expr, Number col-or-expr)`	*Deprecated*. See "Update Built-In Aggregate or Window Analytic Functions for 3.8" on page 769 for alternatives.

Aggregate	Window	Function	Description
✓		`covar(Number col-or-expr, Number col-or-expr)`	Returns the measure of the covariance of the two specified columns or expressions. This measure is not bias corrected. Rows with null values are ignored A positive covariance indicate that the values for these columns tend to vary alike, as one grows larger so does the other. A negative covariance indicates an inverse relationship between the columns.
✓		`covar_pop(Number col-or-expr, Number col-or-expr)`	Returns the full population covariance of the two specified col-or-exprs. See covariance for more information.
✓		`covar_samp(Number col-or-expr, Number col-or-expr)`	Returns the sample covariance (an estimate) of the two specified columns. See covariance for more information.
	✓	`cume_dist()`	The cumulative distribution for the current row in the current partition. This is the percentage of values, including the current row, that are less than or equal to the value for the current row. **Note:** This function requires that partitions be ordered and cannot be applied to windows within the partition.
	✓	`dense_rank()`	Assigns an indexed rank number indicating the order within this partition based on the unique combined values

Aggregate	Window	Function	Description
			for all columns specified in the Order By clause defined for this partition. All rows with the same unique combination receive the same rank. Rank numbers are contiguous across the partition. For an example illustrating this function, see "Example 122. Row_number, Rank and Dense_rank Example" on page 758. See also the `rank()` plain function. **Note:** This function requires that partitions be ordered and cannot be applied to windows within the partition.
	✓	`analytics.discretize(Number col-or-expr, int binCount)`	This function segments a continuous range of values for a column or expression into discrete *bins* based on the number of bins specified. If the value: ■ < = minimum value for the column, it returns 0. ■ > = maximum value for the column, it returns `binCount - 1`. ■ < maximum and > minimum value for the column, it returns a fraction between 0 and `binCount - 1` indicating the value's relative position within the number of bins. So 0 <= return value < 1, indicates the first bin, 1 <= return value < 2, indicates the second bin, and so on.

Aggregate	Window	Function	Description
			Note that this function does not support windows. It can only be applied to the entire partition.
	✓	`first_value(Object col-or-expr)` `first_value(Object col-or-expr, Any default)`	Returns the value for the specified column or expression for the first row in the partition or window for the current row. If a default value is specified, this returns the default value in cases where the specified window is empty.
✓		`gmean(Number col-or-expr)`	Returns the geometric mean of the values for this column or expression within the partition or the current window in this partition. The geometric mean is typically used to define a mean when more than one property is involved, especially if the scale for the properties is different.
✓		`analytics.kmean_clusters(Object col-or-expr1[,Number col-or-expr2,...Number col-or-exprN]; Integer k, Integer iterations, String measure)`	*Deprecated*. See `analytics.kmeans_clusters`.
✓		`analytics.kmeans_clusters(Object col-or-expr1[,Number col-or-expr2,...Number col-or-exprN]; Integer k, Integer`	Returns the optimum center point for the number of clusters specified as *k* that group rows within a minimum distance for each cluster. Inclusion in a cluster for each row is defined by the *features* specified in the list of column

Aggregate	Window	Function	Description
		`iterations, String measure)`	parameters (before the semi-colon). The distance from the cluster center point is measured by the formula specified in *measure* . ■ `col-or-expr1, col-or-expr2, ... col-or-exprN;`: are individual columns or expressions used as features of the vectors that define clusters. This list of parameters *must*: ■ Consist solely of columns with numeric values. ■ Contain at least one column, but can have any number of columns. ■ End with a semi-colon (;). This indicates the end of feature parameters and the beginning of the remaining, well-known parameters. ■ `k`: is the number of clusters to create ■ `iterations`: is the maximum number of iterations to perform to optimize clusters. ■ `measure`: is the name of the formula to use to define membership for each row in a given cluster. Valid measures are: ■ `euclidean` ■ `manhattan` ■ `cosine` ■ `tanimoto` ■ `squaredeuclidean`

Aggregate	Window	Function	Description
✓		`analytics.kmean_observations(Number col-or-expr1[,Number col-or-expr2,...Number col-or-exprN]; Integer k, Integer iterations, String measure)`	*Deprecated.* See `analytics.kmeans_observations`.
	✓	`analytics.kmeans_observations(Number col-or-expr1[,Number col-or-expr2,...Number col-or-exprN]; Integer k, Integer iterations, String measure)`	Returns the ID of the cluster that each row belongs to for the number of clusters specified as *k* . Inclusion in a cluster for each row is defined by the *features* specified in the list of column parameters (before the semi-colon). The distance from the cluster center point is measured by the formula specified in *measure* . ■ `col-or-expr1, col-or-expr2, ... col-or-exprN;`: are individual columns or expressions used as features of the vectors that define clusters. This list of parameters *must*: ■ Consist solely of columns with numeric values. ■ Contain at least one column, but can have any number of columns. ■ End with a semi-colon (;). This indicates the end of feature parameters and the beginning of the remaining, well-known parameters. ■ `k`: is the number of clusters to create

Aggregate	Window	Function	Description
			■ `iterations`: is the maximum number of iterations to perform to optimize clusters ■ `measure`: is the name of the formula to use to define membership for each row in a given cluster. Valid measures are: ■ `euclidean` ■ `manhattan` ■ `cosine` ■ `tanimoto` ■ `squaredeuclidean`

	✓	`lag(Object col-or-expr)`
		`lag(Object col-or-expr, int offset)`
		`lag(Object col-or-expr, int offset, Object default)`
			Returns the value for the specified column or expression from a preceding row for the current row.
			The preceding row is found using the offset specified or using 1 if this parameter is omitted.
			If the specified preceding row is outside the scope of this partition or the current window within this partition, this returns the `default`, if specified, or returns `null`. The data type for `default` must match the data type for `col-or-expr`.

	✓	`last_value(Object col-or-expr)`	Returns the value for this column or expression for the last row in this partition or window.
	✓	`lead(Object col-or-expr)`	Returns the value for the specified column or expression

Aggregate	Window	Function	Description
		`lead(Object col-or-expr, int offset)` `lead(Object col-or-expr, int offset, Object default)`	from a following row for the current row. The following row is found using the offset specified or using 1 if this parameter is omitted. If the specified following row is outside the scope of this partition or the current window within this partition, this returns the `default`, ifspecified, or returns `null`. The data type for `default` must match the data type for `col-or-expr`.
✓		`analytics.linear_regression_intercept(Number col-or-expr1, Number col-or-expr2)`	Returns the intercept value for the line resulting from a linear regression of the two columns or expressions passed as arguments. This uses an ordinary least squares regression where the first column is considered the independent variable.
	✓	`analytics.linear_regression_predicted(Number col-or-expr, Number col-or-expr)`	Returns the expected value (the Y coordinate) for the point on the regression line for each row in this dataset. This line is derived from a linear regression applied to the two specified columns or expressions.
✓		`analytics.linear_regression_slope(Number col-or-expr, Number col-or-expr)`	Returns the slope value for the line resulting from a linear regression of the two columns or expressions passed as arguments. This uses an ordinary least squares regression where the first column is considered the independent variable.

Aggregate	Window	Function	Description
✓		`max(Number col-or-expr)`	Returns the maximum value of this column or expression for all rows within the group, partition or window scope.
✓		`min(Number col-or-expr)`	Returns the minimum value of this column or expression for all rows within the group, partition or window scope.
	✓	`analytics.ordinal(String col-or-expr)`	Returns an index number for each unique string value in the specified column or expression. Typically, assigning an ordinal number allows computations to work with string columns.
	✓	`ntile(Number groups)`	Distributes the rows in an ordered partition into the specified number of groups (the percentiles). **Note:** This function requires that partitions be ordered and cannot be applied to windows within the partition.
	✓	`nth_value(Object col-or-expr, Integer n)` `nth_value(Object col-or-expr, Integer n, Object default)`	Returns the value for the expression or column of the specified row (*n*) in the current partition and window. The `n` parameter must be a positive integer. The `col-or-expr` parameter can be a column or any calculation. If the specified nth row is outside the scope of this partition or the current window within this partition, this returns the `default`, if specified, or returns `null`. The data type for `default` must

Aggregate	Window	Function	Description
			match the data type for `col-or-expr`.
	✓	`percent_rank()`	The percentage of rows that are less than the value for the current row based on the number of rows in the current partition and window not counting the highest value. **Note** This function requires that partitions be ordered and cannot be applied to windows within the partition.
	✓	`rank()`	Assigns an indexed rank number indicating the order within this partition based on the unique combined values for all columns in the Order By clause defined for this partition. All rows with the same unique value receive the same rank. Rank numbers may not be contiguous, as they are skipped if multiple rows have identical ranks. For an example illustrating this function, see "Example 122. Row_number, Rank and Dense_rank Example" on page 758. See also the `dense_rank()` plain function. **Note** This function requires that partitions be ordered and cannot be applied to windows within the partition.
✓		`regr_intercept(Number col-or-expr1,`	See analytics.linear_regression_intercept.

Aggregate	Window	Function	Description
		`Number col-or-expr2)`	
✓		`regr_r2(Number col-or-expr1, Number col-or-expr2)`	Returns the coefficient of determination (R squared) of a linear regression line based on the specified columns or expressions. This indicates how well the linear regression line is fitted to the data. This ranges from 0 to 1, but can return null if there is no variance in the values of *col-or-expr2* .
✓		`regr_slope(Number col-or-expr1, Number col-or-expr2)`	See analytics.linear_regression_slope.
	✓	`row_number()`	Returns a sequential index number for this row within the current partition. For an example illustrating this function, see "Example 122. Row_number, Rank and Dense_rank Example" on page 758. **Note** This function cannot be applied to windows within the partition.
✓		`skew(Number col-or-expr)`	Returns a measure of the asymmetry from a normal probability distribution of the values for this column or expression within the current partition or the current window in this partition. This uses the adjusted Fisher-Pearson standardized moment coefficient.

Aggregate	Window	Function	Description
			Positive values tend to indicate that more values are found left of the mean, while negative values imply the reverse. This returns `0` if there are not at least three rows in the dataset. This function can also return `NaN` if the caluclation is undefined or not a number.
✓		`stddev(Number col-or-expr)`	Returns the standard deviation of the values for this column or expression in the partition or the current window in this partition. Null values are ignored. If there are no numeric values, this returns `NaN`. If there is only a single row, this returns zero.
✓		`stddev_pop(Number col-or-expr)`	Returns the full population standard deviation of the values for this column or expression. See stddev for more information.
✓		`stddev_samp(Number col-or-expr)`	Returns the sample standard deviation (an estimate) of the values for this column or expression. See stddev for more information.
✓		`sum(Number col-or-expr)`	Returns the sum of all values of this column or expression for the rows within the group, partition or window scope. Null values are ignored. If all values are null or empty, return null.

Aggregate	Window	Function	Description
✓		`variance(Number col-or-expr)`	Returns the variance of the distribution of values for this column or expression within the current partition or the current window in this partition. Null values are ignored. If the column has no numeric values, this returns `NaN`.
✓		`var_pop(Number col-or-expr)`	Returns the full population variance for this column or expression. See variance for more information.
✓		`var_samp(Number col-or-expr)`	Returns the sample variance (an esitmate) for the values of this column or expression. See variance for more information.

`Distinct` Aggregations

Standard SQL aggregations functions such as `AVG`, `SUM`, `COUNT`, etc. (see list below) ignore duplicate values if their parameters are given in conjunction with the `DISTINCT` key word.

`DISTINCT` may be used with `AVG`, `COUNT`, `MAX`, `MIN`, `STDDEV`, `STDDEV_POP`, `STDDEV_SAMP`, `SUM`, `VARIANCE`, `VAR_POP`, `VAR_SAMP` and `COUNT`.

`DISTINCT` may however not be used with `COVAR`, `COVAR_POP`, `COVAR_SAMP`, `CORR`, `REGR_SLOPE`, `REGR_INTERCEPT` and any user-defined aggregation function.

Table 1. Example

aColumn
1
2
2
3

aColumn
3
3

Then the query **SELECT SUM(DISTINCT aColumn) FROM source** only sums up the unique values of the column **aColumn** and returns **6**.

Window functions over RANGEs

In RAQL`SELECT aggregationFunction(x) OVER(ORDER BY y) FROM s` is semantically equivalent to `SELECT aggregationFunction (x) OVER(ORDER BY y ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM s.` The SQL Standard however defines the semantics to correspond with `SELECT aggregationFunction (x) OVER(ORDER BY y RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM s` which means that not only the current row but all "peer" rows (rows identical to the current row with respect to the ordering) are contained in the window. That means, that a rolling aggregate without an explicit window frame specification currently yields different results in RAQL and any SQL standard compliant DBMS.

In order to ensure a high level of compliance with the SQL standard, window frames are also allowed to be specified using the `RANGE` key word.

Semantics

Window frames defined using the `ROWS` keyword are easy to understand. The window frame clause `ROWS BETWEEN x PRECEDING AND y FOLLOWING` simply defines two offsets **x** and **y** that for each row in the input partition define the first and the last row number of the corresponding window relative to the current row number. `PRECEDING` and `FOLLOWING` only indicate whether the offset is negative (the row number precedes the current row number) or positive respectively.

Table 2. Example

rowNumber	ROWS BETWEEN 3 PRECEDING AND 3 FOLLOWING	ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING	ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
1	Window = [1,4]	Window = []	Window = [1,10]
2	Window = [1,5]	Window = [1,1]	Window = [2,10]

rowNumber	ROWS BETWEEN 3 PRECEDING AND 3 FOLLOWING	ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING	ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
3	Window = [1,6]	Window = [1,2]	Window = [3,10]
4	Window = [1,7] (from row number 4-3 to 4+3)	Window = [1,3]	Window = [4,10]
5	Window = [2,8]	Window = [2,4]	Window = [5,10]
6	Window = [3,9]	Window = [3,5]	Window = [6,10]
7	Window = [4,10]	Window = [4,6]	Window = [7,10]
8	Window = [5,10]	Window = [5,7]	Window = [8,10]
9	Window = [6,10]	Window = [6,8]	Window = [9,10]
10	Window = [7,10]	Window = [7,9]	Window = [10,10]

On the other hand the semantics of window frame specifications using the `RANGE` keyword does not depend on the physical row numbers but on the actual values of a column. Still, the window frame clause defines two offsets **x** and **y**, but these offsets are not added to the row number but to an actual column value. Hence, any window frame specification involving `RANGE` and an offset other than `UNBOUNDED PRECEDING`, `UNBOUNDED FOLLOWING` or `CURRENT ROW` requires the window to be ordered on a single column and this column needs to have a numeric data type (in order to allow for offset arithmetics).

Table 3. Example

rowNumber	salary	SELECT avg(salary) OVER (ORDER BY salary RANGE BETWEEN 300 PRECEDING AND 300 FOLLOWING) FROM ...	considered window	
1	1750	1850	[1450,2050]	salaries between 1750 - 300 and 1750 + 300
2	1900	1850	[1600,2200]	
3	1900	1850	[1600,2200]	
4	2200	2100	[1900,2500]	
5	2400	2300	[2100,2700]	
6	2750	2866,67	[2450,3050]	
7	2900	2980	[2600,3200]	
8	2950	2980	[2650,3250]	
9	3100	2980	[2800,3400]	
10	3200	2980	[2900,3500]	

Table 4. Example

rowNumber	salary	SELECT avg(salary) OVER (ORDER BY salary RANGE BETWEEN 300 PRECEDING AND 300 FOLLOWING) FROM ...	considered window	
1	1750	1	[1750,1750]	salaries between 1750 - 0 and 1750 + 0
2	1900	2	[1900,1900]	
3	1900	2	[1900,1900]	
4	2200	1	[2200,2100]	
5	2400	1	[2400,2400]	
6	2750	1	[2750,2750]	
7	2900	1	[2900,2900]	
8	2950	1	[2950,2950]	
9	3100	1	[3100,3100]	
10	3200	1	[3200,3200]	

Note: The offsets in a `RANGE` frame clause define logical offsets in the ordered sequence defined by the order by clause. If the order by clause specifies a descending order (see next example) the offset **10 PRECEDING** translates to "a value that precedes the current value in the current sequence and differs at most by 10 (years/units/...)"

Table 5. Example

rowNumber	age	SELECT COUNT(*) OVER (ORDER BY age DESC RANGE 10 PRECEDING) FROM ...	considered window	
1	75	1	[85, 75]	ages between 75 + 10 and 75 - 0
2	68	2	[78, 68]	
3	68	2	[72, 62]	
4	56	2	[66, 56]	
5	55	3	[65, 55]	
6	51	3	[61, 51]	
7	48	5	[58, 48]	
8	48	5	[58, 48]	
9	47	6	[57, 47]	
10	43	5	[53, 43]	

Affected Queries

As mentioned above, the new `RANGE` specifier was introduced to ensure compliance with the SQL standard. There are however only very few cases where existing queries should be affected by those changes. Only queries specifying a window function with

1. an explicit order-by specification and
2. NO explicit window frame specification may be affected if
3. The expressions defined in the order-by clause are not unique for each row.

If the order-by expressions are unique, than for each row there are no peers with respect to the ordering and the implicitly added window frame clause `RANGE UNBOUNDED PRECEDING` is equivalent to `ROWS UNBOUNDED PRECEDING` and hence the query results

would be the same as before. If however the order-by specification is not unique then there may be rows having one or more peer rows so that the query result differs from the result in former RAQL releases and the user might want to explicitly add the window frame clause `ROWS UNBOUNDED PRECEDING` to yield the former results.

Create and Add User-Defined Functions for RAQL Queries

You can define your own functions to use in RAQL queries in addition to the "Built-In RAQL Functions" on page 775. User-defined functions are Java classes that you write, configure and deploy to MashZone NextGen.

User-defined function can be:

- **Plain functions**: used in Select, Over, Where, Order By and Group By clauses in RAQL queries. They typically either *cast* (change) the datatype of column values, extract part of the values or transform values in some way.

  Plain functions are applied individually to each value in the column specified *without* access to values in other rows.

- **Window analytic functions**: use some of the rows in a partition or window to perform a calculation and return a value for each row. Typically, this uses rows that are relative to the current row, such as `first_value` or `row_number`.

  You must include an Over clause in queries that use window analytic functions. This defines the partitions or windows used by the function.

- **Aggregate analytic functions**: use all rows in the dataset, group, partition or window to perform a calculation and return a single value. For example, `sum` adds the values of all rows in the current scope.

  Aggregate analytic functions can be be used in the Select or Having clauses of RAQL queries:

  - To return a single value for each group defined in a Group By clause.
  - To return a single value for each partition or each window defined in an Over clause. This single value is added to each row in the partition or window. You can also use aggregate analytic functions to return running calculations for partitions or windows defined in an Over clause.
  - To return a single value for the entire dataset if no group definition or partition definition is specified.

To write user-defined functions for RAQL, you should "Set Up Your Development Environment" on page 805. Then:

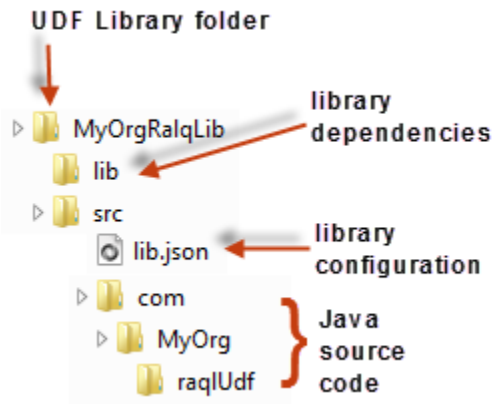
- Write Plain Functions for RAQL
- Write Window Analytic Functions for RAQL
- Write Aggregate Analytic Functions for RAQL
- Use a Factory for Function Overloading, if needed

- Configure, Compile, Deploy and Test User-Defined Functions

Set Up Your Development Environment

To get started, you need:

- The latest Java Development Kit 7. See JDK1.7 to download and install this, if needed.
- A folder for your user-defined function *library* with this structure:



User-defined functions are packaged and deployed in named libraries. The library name also uniquely defines your user-defined functions from built-in functions or user-defined functions in other libraries that have been deployed in MashZone NextGen. Library names must match the name of the library folder containing your source code.

Valid library names must be unique for a MashZone NextGen Server. They must start with a letter and can contain letters, numbers or underscores (_).

Important: The name `analytics` is *reserved* for the MashZone NextGen UDF function library. See "External UDF Library Deployment Folder" on page 810 for more information.

You can organize user-defined functions however you need. User-defined functions can be packaged in multiple libraries. Each library can contain one or more Java packages. Each package can contain multiple classes. For plain functions, each class can contain one or more functions. For analytic functions, each function is packaged as a class.

Note: You *cannot* use periods (.) in either library or user-defined function names as this character is the reserved separator between library and function names. See also "Valid Names for Datasets, Columns, Aliases, Paths and Functions" on page 727 for other function name restrictions.

- JAR files for any third-party Java libraries that your user-defined functions depend on, placed in the *library-name* /lib folder.
- Update your classpath to include the following JAR files:
  - *web-apps-home* /presto/WEB-INF/lib/jackbe-presto-raql-*version* .jar for RAQL.
  - *web-apps-home* /presto/WEB-INF/lib/rtm-*version* .jar for the UDF interfaces.
  - And any third party JAR files you add to *library-name* /lib.

  For an example, see "Configure, Compile, Deploy and Test User-Defined Functions" on page 807.

  Important: It is *not* a good practice to copy either the RAQL or UDF interface JAR files to your *library-name* /lib folder to simplify the classpath. This can cause errors when you deploy your user-defined function library. See "Configure, Compile, Deploy and Test User-Defined Functions" on page 807 for more information.

Before you compile, deploy and test your user-defined functions, you will also need a library configuration file (lib.json). We will cover these requirements later, in "Configure, Compile, Deploy and Test User-Defined Functions" on page 807.

Write Plain Functions for RAQL

Effective in version 3.8, plain functions implement the `UserDefinedFunctionAdapter` interface in the `de.rtm.push.adapters` package of theMashZone NextGenRAQL User Defined Function API. If functions need to support multiple signatures, you can also implement the `UserDefinedFunctionAdapterFactory` interface.

Note: These interfaces are not backwards compatible with the UDF API from MashZone NextGen 3.7 or earlier.

You can implement plain functions more simply as Java classes and let RAQL automatically derive the methods required by the interface. Thus at a minimum plain functions must:

- Import the `com.jackbe.jbp.raql.udx.loader.RaqlFunc` annotation interface.
- Include annotations to identify which methods are RAQL functions.
- Be implemented as public static class methods that return primitive types.

This example contains two plain functions, `replace` and `capitalize`:

```
package com.jackbe.jbp.raql.samplelib.annotated.udf;
/*
 * Sample class in a library of plain user-defined functions for RAQL
 */
import com.jackbe.jbp.raql.udx.loader.RaqlFunc;
public class StringFunctions {
@RaqlFunc
    public static String replace(String val, String oldStr, String newStr) {
    if (val == null || oldStr == null || newStr == null)
```

```
    return null;
    return val.replace(oldStr, newStr);
    }
@RaqlFunc(name="capitalize")
    public static String upper(String val) {
    return val == null ? null : val.toUpperCase();
    }
}
```

The `@RaqlFunc` annotation identifies which methods should be associated with RAQL functions. If you omit the `name` parameter, the name of the method becomes the name of the RAQL function within this UDF library. Use `(name="`*alias*`")` to use a different name for the function from the method name.

Also, plain user-defined functions may be annotated using the `@RaqlFunc` annotation. If within one single class multiple static methods with distinct signatures are annotated with the `@RaqlFunc` annotation and are given the same name, they will all be registered under that name and the engine will choose the most suitable signature for a particular call.

Tip: Using annotations to configure the methods for user-define functions in RAQL is a best practice. However, you can skip the annotations in your Java classes and instead provide configuration that maps methods to user-defined functions in the `lib.json` configuration file for your UDF libraries. See "External UDF Library Deployment Folder" on page 810 for more information.

With plan functions, you can annotate several static methods with different signatures in a single class using the same function name. All of the methods are registered under that single function name. At runtime, the method with the appropriate signature is used.

Configure, Compile, Deploy and Test User-Defined Functions

To compile and deploy user-defined functions

1. Add a folder with the name of your library under the default external UDF library deployment folder *MashZoneNG-install* /raql-udfs.

   Note: The folder for UDF library deployment is set in the system property `-Dpresto.raql.udf.libsDir`. This may be a different folder in clustered environments with a shared external configuration folder for MashZone NextGen.

   This new folder is the root deployment folder for your library. For example:

   /SoftwareAG/MashZoneNG/raql-udfs/MyOrgRaqlLib

2. Complete configuration that identifies the Java packages for this library in a lib.json file. This file must reside in the root deployment folder for your library, created in the previous step.

Note: You may also want to place a copy of this configuration in your library development folder to track in your source control system.

This file uses the JSON format to identify packages that contain user-defined functions for a library. For example:

```
{
    "exportedFuncs" : {
  "annotatedPackages" : [
   "com.jackbe.jbp.raql.samplelib.annotated.udf",
   "com.jackbe.jbp.raql.samplelib.annotated.uda",
   "com.jackbe.jbp.raql.samplelib.annotated.udw"
  ],
        "classes" : [
            {
                "name" : "com.jackbe.jbp.raql.samplelib.configured.MoreStringFunctions",
                "funcs" : [
                    { "name" : "low", "method" : "lower" },
     { "method" : "hashCode" }
                ]
            },
   {
                "name" : "com.jackbe.jbp.raql.samplelib.configured.AverageAggregationFunctionI
                "funcs" : [
                 { "name" : "avg" }
             ]
            },
            {
                "name" : "com.jackbe.jbp.raql.samplelib.configured.LeadWindowFunction",
                "funcs" : [
                 { "name" : "lead" }
             ]
            }
        ]
    }
}
```

This example identifies three packages for this library. The `classes` property must be present in configuration, but can be an empty array like this example. The RAQL Engine will search each class within the identified packages for function annotations to find the functions to add.

You can also optionally identify classes, functions and methods in this configuration. See "UDF Library Configuration" on page 809 for more information on this method.

3. Compile your Java class with user-defined functions, being sure to include the jackbe-presto-raql-*version* .jar and rtm-*version* .jar files and any third-party libraries you used in this class in the classpath.

   - For Windows for example:

     ```
     %JAVA_HOME%/bin/javac -classpath c:\SoftwareAG\MashZoneNG\apache-tomee-jaxrs\webapps\prest
     c:\SoftwareAG\MashZoneNG\apache-tomee-jaxrs\webapps\presto\WEB-INF\lib\rtm-9.7.0.jar
     -d classes src\com\MyOrg\raqlUdf\MyReplaceFunc.java
     ```

   - For Linux, OS/X or UNIX for example:

     ```
     javac -classpath /SoftwareAG/MashZoneNG/apache-tomee-jaxrs/webapps/presto/WEB-INF/lib/jack
     /SoftwareAG/MashZoneNG/apache-tomee-jaxrs/webapps/presto/WEB-INF/lib/rtm-9.7.0.jar
     -d classes src/com/MyOrg/raqlUdf/MyStringFuncs.java
     ```

This will add the compiled class to the `classes` folder in your development folder for this library.

4. Deploy the classes and any third-party libraries for this library to the MashZone NextGen Server. Copy the following folders to the root deployment folder for your library:
   - /classes folder and all contents
   - /lib folder and all contents

   Important: If you have included either the RAQL or RTM JAR files in your *library-name* /lib folder, do *not* copy these JARs to the deployment folder for your library as this causes errors when functions in the library are used.

5. Restart the MashZone NextGen Server.
6. Write mashups that use these new user-defined functions to test them.

   You refer to user-defined functions in the form *library-name .function-name(arg[,...)* to identify both the library name and the function name. For example:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML
/../schemas/EMMLPrestoSpec.xsd'
xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
name='newMashup'>

  <output name='result' type='document' />
  <variable name='plants' type='document' subtype='csv'
stream='true'
  datafile='mfgplants.csv' />

  <raql outputvariable='result'>
    select Country, Name, MyOrgRaqlLib.replace(Name,'AT','XY')
as Changed from plants
  </raql>
</mashup>
```

UDF Library Configuration

The lib.json file contains configuration that identifies the Java packages with user-defined functions for RAQL. It can also optionally contain configuration that identifies the specific classes and methods in these packages and the function names to map to methods.

Note: It is a best practice to configure methods and function names using annotations in the Java classes directly. If you choose to use lib.json configuration instead, you should omit the annotations in your Java classes for user-defined functions.

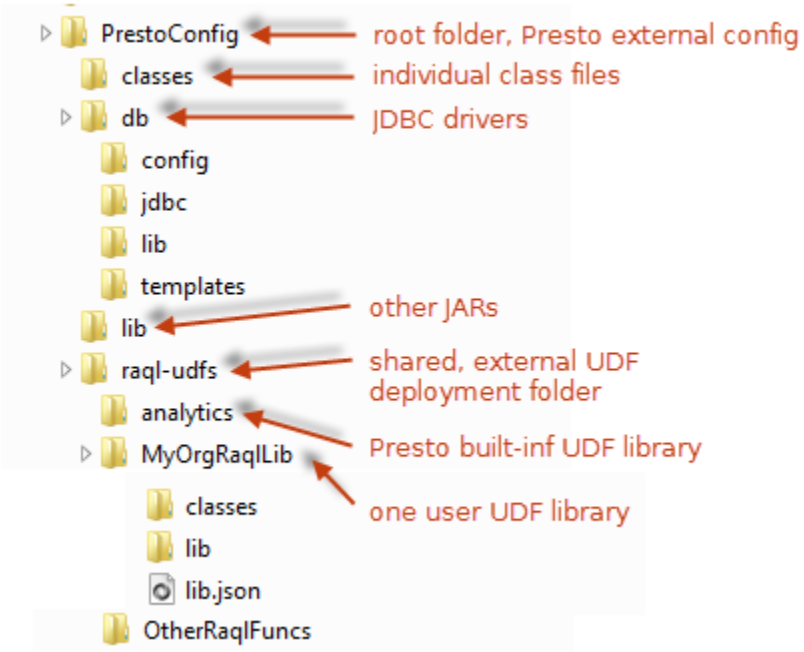
The following example includes method and function mapping configuration along with the required package configuration information:

```
{
  "exportedFuncs": {
    "annotatedPackages":[
      "com.MyOrg.raqlUdf",
      "com.MyOrg.aggregate.raqlUdf",
      "com.MyOrg.window.raqlUdf" ],
    "classes":[
      { "name": "com.MyOrg.raqlUDF.MyStringFuncs",
        "funcs":[ {"method": "replace" },
          { "name": "capitalize", "method": "upper"} ] },
      { "name": "com.MyOrg.aggRaqlUdf.KurtosisFunction",
        "funcs": [ {"name": "kurtosis" ] }
    ]
  }
}
```

External UDF Library Deployment Folder

A default external UDF Library Deployment folder is created when you install MashZone NextGen at *MashZoneNG-install* /raql-udfs where you can deploy all your user-defined functions.

In clustered environments, you may want to create a shared external folder for MashZone NextGen configuration and *move all* user-defined functions, including the MashZone NextGen built-in function library, to this shared location for all members of the cluster. The structure would look something like this:



If you move user-defined functions from the default UDF Library Deployment folder, you must also update an environmental variable for each MashZone NextGen Server:

1. Edit the script for the appropriate operating system in any text editor of your choice:
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat file, for Windows systems, or
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX systems.
2. Add or update the `-Dpresto.raql.udf.libsDir` system property with the path to point to the new shared location.
3. Save your changes and restart the MashZone NextGen Server.

Write Window Analytic Functions for RAQL

With window analytic functions, each function is a single class that implements the `UserDefinedWindowFunctionAdapter` interface (`de.rtm.push.adapters.windowfunctions` package) from theMashZone NextGenRAQL User Defined Function API.

Window analytic functions, unlike plain functions, have access to all rows, or *records* within the current partition or window which they can use to perform calculations. Unlike aggregate analytic functions, however, they provide a different calculation for each record.

To accomplish this, window analytic functions use the following methods:

- `createInitialState()`: to reset state for the current window or partition.

- `checkWindowSpecificiation(boolean isPartitioned, boolean isOrdered, WindowFrameSpecification windowFrameSpec)`: to validate that the window definition meets requirements such as being sorted.
- `call(S state, UserDefinedWindowFunctionAdapter.partitionEntry currentEntry, UserDefinedWindowFunctionAdapater.partion partition, int currentIndex, WindowFrame currentWindowSpec, WindowFrame prevWindowSpec)`: window analytic functions must implement this method with the core logic of the function and return the result of the calculation as `WindowFunctionResult<S,O>`.

There are also basic 'housekeeping' methods: `getParameterTypes()` and `getReturnType()`.

You set up your development environment for window analytic functions just the same as for plain functions. See "Set Up Your Development Environment" on page 805 for details.

We're going to use two examples. The first example shows the basics of a window analytics function and how to track state and set the function result. This example implements a simple sum. To create this function, you:

1. Construct and Initialize the Window Analytic Function Class
2. Implement the call Method for the Window Analytic Function

And Configure, Compile, Deploy and Test User-Defined Functions. For the complete code, see "Example 124. Complete WindowSum Example" on page 814.

The second example implements the MashZone NextGen built-in `lead` function which illustrates techniques to "Example 125. Work with Specific Records in Window Calculations" on page 816 using the current position of a record.

Construct and Initialize the Window Analytic Function Class

This example, MySumWindowFunction.java is available in the sample user-defined functions package at *MashZoneNG-install* /raql-udfs/SampleRaqlLib.

Your window analytic function class imports the RAQL UDF annotation class, `com.jackbe.jbp.raql.udx.loader.RaqlFunc`, various classes in theMashZone NextGenRAQL User Defined Function API and implements the `UserDefinedWindowFunctionAdapter` interface:

```
package com.raql.samples;
import java.util.Arrays;
import com.jackbe.jbp.raql.udx.loader.RaqlFunc;
import de.rtm.push.adapters.Adapters;
import de.rtm.push.adapters.windowfunctions.UserDefinedWindowFunctionAdapter;
import de.rtm.push.adapters.windowfunctions.WindowFrame
import de.rtm.push.adapters.windowfunctions.WindowFrameSpecification;
import de.rtm.push.adapters.windowfunctions.WindowFunctionResult;
import de.rtm.util.exception.IncompatibleWindowSpecificationException;
import de.rtm.util.records.metadata.FieldMetaData;
import de.rtm.util.records.metadata.FieldMetaDatas;
import de.rtm.util.records.types.JavaTypes;
import de.rtm.util.records.types.JavaTypes.Type;
/**
```

```
 * This window function adapter computes the sum of a window.
 */
@RaqlFunc(name="mySumFunction")
public static class WindowSum implements UserDefinedWindowFunctionAdapter<Double, Double> {
  protected final Type[] parameterTypes;
  public WindowSum(Type[] parameterTypes) {
    this.parameterTypes = parameterTypes;
  }
  /**
   * Returns the type of the result, which is simply always Double.
   */
  @Override  public Type getReturnType() {
    return Type.DOUBLE;
  }
  /**
   * Returns the types of the input parameters.
   */
  @Override  public Type[] getParameterTypes() {
    return parameterTypes;
  }
  /**
   * Creates the initial state of the window sum, which is 0.
   */
  @Override
  public Double createInitialState() {
    return 0d;
  }
}
```

This example overrides the default implementation for `createInitialState` to reset the sum to zero.

Implement the call Method for the Window Analytic Function

You implement the `call` method with the core logic for your window anlaytic function. The `state` parameter represents either the initial state (for the first window evaluation) or the state computed from the previous window evaluation.

Note: User-defined functions should be stateless wherever possible as a best practice. Instead, you can manage intermediate state information for window calculations in the `state` property of the `WindowFunctionResult`.

It also has parameters for the current row within the current partition, the current partition, the index for the current row and specifications for the current window and the previous window that define the context for the function.

```
...
  /**
   * Computes the sum for the current window. Instead of simply summing
   * up the values of the current window, only the required values
   * from current and previous window are combined with the state of
   * the previous window evaluation. This approach allows for a more
   * efficient evaluation.
   *
   * @param state   The state which is either the initial state for
   *                the first window evaluation or the state as
   *                computed during the previous window evaluation
   * @param currentEntry  The current row of the partition being
   *                      processed
   * @param partition  The partition being processed
```

```
   * @param currentIndex  The index of the current row being
   *                      processed
   * @param currentWindowSpec  The specification of the current window
   *                           frame within the partition
   * @param previousWindowSpec  The specification of the previous window
   *                           frame within the partition
   * @return  The result of the window function for the current window
   */
  @Override
  public WindowFunctionResult<Double, Double> call(Double state, PartitionEntry currentEntry,
  Partition partition, int currentIndex, WindowFrame currentWindowSpec, WindowFrame previousWindo
    // initialize new sum with old sum from state
    Double newSum = state;
    // if previous window has values, remove any from the subtotal that
    //  are not in current window
    boolean valuesInPreviousWindow = previousWindowSpec != null && !previousWindowSpec.isEmpty();
    if (valuesInPreviousWindow) {
      final int removeTo = Math.min(currentWindowSpec.getStartIndex(), previousWindowSpec.getEndI
      for (int i = previousWindowSpec.getStartIndex(); i < removeTo; i++) {
        newSum -= ((Number) partition.get(i).getColumnValue(0)).doubleValue();
      }
    }
    // and add values from current window that are not in previous window
    final int addFrom = !valuesInPreviousWindow?currentWindowSpec.getStartIndex()
    :Math.max(currentWindowSpec.getStartIndex(), previousWindowSpec.getEndIndex());
    for (int i = addFrom; i < currentWindowSpec.getEndIndex(); i++) {
      newSum += ((Number)partition.get(i).getColumnValue(0)).doubleValue();
    }
    // return the new sum as state for use in the next window
    // evaluation and as result for the current row
return Adapters.createWindowFunctionResult(newSum, newSum);
  }
  @Override
  public void checkWindowSpecification(boolean isPartitioned, boolean isOrdered, WindowFrameSpeci
  windowFrameSpecification) throws IncompatibleWindowSpecificationException {
    // nothing to be done
  }
}
```

The `call` method returns an object that implements the `WindowFunctionResult` interface. `WindowFunctionResult` instances contain:

- The result of the function, in this case the sum of this numeric column for the current row in the current window and partition.
- State information needed to apply this function to the next row.

  In this example, the state is also simply the result of the function. The calculation of the sum for each row uses this as a starting point and then backs out values for any rows that are no longer considered part of the window and adds in values for any new rows in the window.

Lastly, this implements the `checkWindowSpecification` method that is required for the interface as a no op method.

Complete WindowSum Example

```
package com.raql.samples;
import  java.util.Arrays;
import  com.jackbe.jbp.raql.udx.loader.RaqlFunc;
import de.rtm.push.adapters.Adapters;
```

```
import de.rtm.push.adapters.windowfunctions.UserDefinedWindowFunctionAdapter;
import de.rtm.push.adapters.windowfunctions.WindowFrame
import de.rtm.push.adapters.windowfunctions.WindowFrameSpecification;
import de.rtm.push.adapters.windowfunctions.WindowFunctionResult;
import de.rtm.util.exception.IncompatibleWindowSpecificationException;
import de.rtm.util.records.metadata.FieldMetaData;
import de.rtm.util.records.metadata.FieldMetaDatas;
import de.rtm.util.records.types.JavaTypes;
import de.rtm.util.records.types.JavaTypes.Type;
/**
 * This window function adapter computes the sum of a window.
 */
@RaqlFunc(name="mySumFunction")
public static class WindowSum implements UserDefinedWindowFunctionAdapter<Double, Double> {
  protected final Type[] parameterTypes;
  public WindowSum(Type[] parameterTypes) {
    this.parameterTypes = parameterTypes;
  }
  /**
   * Returns the type of the result, which is simply always Double.
   */
  @Override  public Type getReturnType() {
    return Type.DOUBLE;
  }
  /**
   * Returns the types of the input parameters.
   */
  @Override  public Type[] getParameterTypes() {
    return parameterTypes;
  }
  /**
   * Creates the initial state of the window sum, which is 0.
   */
  @Override
  public Double createInitialState() {
    return 0d;
  }
  /**
   * Computes the sum for the current window. Instead of simply summing
   * up the values of the current window, only the required values
   * from current and previous window are combined with the state of
   * the previous window evaluation. This approach allows for a more
   * efficient evaluation.
   *
   * @param state   The state which is either the initial state for
   *                the first window evaluation or the state as
   *                computed during the previous window evaluation
   * @param currentEntry  The current row of the partition being
   *                      processed
   * @param partition  The partition being processed
   * @param currentIndex  The index of the current row being
   *                      processed
   * @param currentWindowSpec  The specification of the current window
   *                           frame within the partition
   * @param previousWindowSpec  The specification of the previous window
   *                           frame within the partition
   * @return  The result of the window function for the current window
   */
  @Override
  public WindowFunctionResult<Double, Double> call(Double state, PartitionEntry currentEntry,
  Partition partition, int currentIndex, WindowFrame currentWindowSpec, WindowFrame previousWindo
    // initialize new sum with old sum from state
    Double newSum = state;
```

```
    // if previous window has values, remove any from the subtotal that
    //  are not in current window
    boolean valuesInPreviousWindow = previousWindowSpec != null && !previousWindowSpec.isEmpty();
    if (valuesInPreviousWindow) {
      final int removeTo = Math.min(currentWindowSpec.getStartIndex(), previousWindowSpec.getEndI
      for (int i = previousWindowSpec.getStartIndex(); i < removeTo; i++) {
        newSum -= ((Number) partition.get(i).getColumnValue(0)).doubleValue();
      }
    }
    // and add values from current window that are not in previous window
    final int addFrom = !valuesInPreviousWindow?currentWindowSpec.getStartIndex()
    :Math.max(currentWindowSpec.getStartIndex(), previousWindowSpec.getEndIndex());
    for (int i = addFrom; i < currentWindowSpec.getEndIndex(); i++) {
      newSum += ((Number)partition.get(i).getColumnValue(0)).doubleValue();
    }
    // return the new sum as state for use in the next window
    // evaluation and as result for the current row
    return Adapters.createWindowFunctionResult(newSum, newSum);
  }
  @Override
  public void checkWindowSpecification(boolean isPartitioned, boolean isOrdered,
  WindowFrameSpecification windowFrameSpecification) throws IncompatibleWindowSpecificationExcept
    // nothing to be done
  }
}
```

Work with Specific Records in Window Calculations

This example is a window analytics function similar to the MashZone NextGen built-in `lead` function which returns the value for the specified column for a row (a record) that follows the current row by a specific offset:

```
package com.raql.samples;
import java.io.Serializable;
import com.jackbe.jbp.raql.udx.loader.RaqlFunc;
import de.rtm.push.adapters.Adapters;
import de.rtm.push.adapters.windowfunctions.UserDefinedWindowFunctionAdapter;
import de.rtm.push.adapters.windowfunctions.WindowFrame;
import de.rtm.push.adapters.windowfunctions.WindowFrameSpecification;
import de.rtm.push.adapters.windowfunctions.WindowFunctionResult;
import de.rtm.util.exception.IncompatibleWindowSpecificationException;
import de.rtm.util.records.metadata.FieldMetaData;
import de.rtm.util.records.metadata.FieldMetaDatas;
import de.rtm.util.records.types.JavaTypes.Type;
/**
 * This window function adapter determines the lead value for a
 * specific row based on an offset within a window.
 */
@RaqlFunc(name="myLeadFunction")
public static class Lead implements UserDefinedWindowFunctionAdapter<Serializable, Serializable>
  private final static Serializable DEFAULT_DEFAULT_VALUE = null;
  private final static int DEFAULT_OFFSET = 1;
  protected final Type returnType;
  protected final Type[] parameterTypes;
  public Lead(Type[] parameterTypes) {
    this.returnType = parameterTypes[0];
    this.parameterTypes = parameterTypes;
  }
  /**
   * Returns the type of the result.
   */
  @Override
  public Type getReturnType() {
```

```
    return returnType;
  }
  /**
   * Returns the types of the input parameters.
   */
  @Override
  public Type[] getParameterTypes() {
    return parameterTypes;
  }
  /**
   * Creates the initial state.
   */
  @Override
  public Serializable createInitialState() {
    return null;
  }
  /**
   * Determines the lead value of the current window. This has two
   * optional parameters: the offset and the default value ?where?.
   *
   * @return  The result of the window function for the current window
   */
  @Override
  public WindowFunctionResult<Serializable, Serializable> call(Serializable state,
  PartitionEntry currentEntry, Partition partition, int currentIndex, WindowFrame currentWindowSp
  WindowFrame previousWindowSpec) {
    final Serializable[] columnValues = currentEntry.getColumnValues();
    // when the offset is not specified, use the default offset, which is 1
    final int offset = columnValues.length > 1 ? (Integer)columnValues[1] : DEFAULT_OFFSET;
    final int index = currentIndex + offset;
    final Serializable result;
    if (index < currentWindowSpec.getStartIndex() || index >= currentWindowSpec.getEndIndex())
      // when the default value is not specified, use the default default value, which is null
      result = columnValues.length > 2 ? columnValues[2] : DEFAULT_DEFAULT_VALUE;
    else
      result = partition.get(index).getColumnValue(0);
      return Adapters.createWindowFunctionResult(result, result);
  }
  @Override
  public void checkWindowSpecification(boolean isPartitioned, boolean isOrdered, WindowFrameSpeci
  windowFrameSpecification) throws IncompatibleWindowSpecificationException {
    // nothing to be done
  }
}
```

To work with a specific record relative to the current record, this function uses both the index of the current record as well as the specification of the current window.

Write Aggregate Analytic Functions for RAQL

Aggregate analytic functions are a single class that implement the `UserDefinedAggregateFunctionAdapters` interface in the `de.rtm.push.adapters` package of theMashZone NextGenRAQL User Defined Function API.

Like window analytic functions, aggregate analytic functions have access to all rows, or *records* in the current group, partition or window. They perform an aggregate calculation that uses all records to return a single result. Depending on where they are used in RAQL queries, this single result may be all the query returns or it may be returned as a column on every row.

To accomplish this, aggregate analytic functions use the following methods:

- `createInitialState()`: is optional. This resets the aggregate state for the current window, partition or group to whatever value or state the caluclation should begin from.
- `aggregate(S state, Serializable[] values)`: performs an intermediate calculation using the current state and the value of the next record to obtain a new intermediate state.
- `getAggregate(S state)`: implements the final computation for the group, partition or window, if any, and returns the final aggregate result once all intermediate calculations are complete.

There are also several 'housekeeping' methods you must implement: `getAggregateType()`, `getParameterTypes()`, `isEmptyAggregate(S state)`, `negativeCall(S state, Serializable[] values)` and `supportsPN()`.

You set up your development environment for aggregate analytic functions just the same as for plain functions. See "Set Up Your Development Environment" on page 805 for details. Write the class for your aggregate analytic function and then Configure, Compile, Deploy and Test User-Defined Functions.

We're going to use two examples. The first example, Example 126. My Average Aggregate Example is an implementation of the built-in `avg(Number column)` method. This example shows the basics of an aggregate analytics function, how to track state for intermediate steps and then perform the final calculation.

The second example, Example 127. Kurtosis Using a Third Party Library, uses methods in the Apache Commons Math library to calculate the kurtosis for a column. This is an example of how to use third-party libraries.

My Average Aggregate Example

This example, MyAverageAggregationFunction is available in the sample user-defined functions package at *MashZoneNG-install* /raql-udfs/SampleRaqlLib. It implements an aggregate function `myAverageFunction(Number column)` similar to the MashZone NextGen built-in average function:

The `aggregate` method performs intermediate calculations. This increments the number of rows processed and adds the current column value to a subtotal. Both of these intermediate values are added to the `state` object. The `getAggregate` method then uses `state` to calculate the group/partition/window average.

```
package com.raql.samples;
import java.io.Serializable;
import java.util.Arrays;
import com.jackbe.jbp.raql.udx.loader.RaqlFunc;
import de.rtm.push.adapters.DefaultSignature;
import de.rtm.push.adapters.UserDefinedAggregationFunctionAdapter;
import de.rtm.util.exception.UserDefinedAggregationFunctionException;
import de.rtm.util.records.metadata.FieldMetaData;
import de.rtm.util.records.metadata.FieldMetaDatas;
import de.rtm.util.records.types.JavaTypes;
import de.rtm.util.records.types.JavaTypes.Type;
/**
```

```
 * This aggregate adapter computes the average of a group, partition
 * or window.
 */
@RaqlFunc(name="myAverageFunction")
public static class MyAverageAdapter implements UserDefinedAggregationFunctionAdapter<Number[]> {
  /**
   * Returns the types of the input parameters.
   */
  @Override
   public Type[] getParameterTypes() {
     return new Type[] {Type.DOUBLE};
   }
  /**
   * Returns the type of the result.
   */
  @Override
  public Type getAggregateType() {
    return Type.DOUBLE;
  }
  /**
   * Creates the initial state of the average.
   */
  @Override
  public Number[] createInitialState() {
    // the first value is the count and the second the cumulative sum
    return new Number[] {0l, 0d};
  }
  /**
   * Aggregates the current internal state with the new input values
   * and derives a new internal state.
   * @param state  The current internal state
   * @param values  The new input values
   * @return  The new internal state
   */
  @Override
  public Number[] aggregate(Number[] state, Serializable[] values) {
    Long n = (Long) state[0];
    Double sum = (Double) state[1];
    // null-aware handling of input values
    if (values[0] != null) {
      sum += (Double) values[0];
      n += 1;
    }
    // always return a new state
    return new Number[] {n, sum};
  }
  /**
   * Derives the final aggregate value from the current internal state.
   * @param state the internal state
   * @return the aggregate value
   */
  @Override
  public Object getAggregate(Number[] state) {
    long n = (Long) state[0];
    double sum = (Double) state[1];
    // the average of an empty data set is null
    if (n == 0l)
      return null;
    return sum/n;
  }
  /**
   * Removes the input values from the internal state.
   * This step is required for the support of the
```

```
   * positive/negative approach.
   * @param state the current internal state
   * @param values the input values
   * @return the new internal state
   */
  @Override
  public Number[] negativeCall(Number[] state, Serializable[] values) {
    if (values[0] == null)
      return state;
    long n = (Long) state[0];
    double sum = (Double) state[1];
    n--;
    sum -= (Double) values[0];
    // always return a new state
    return new Number[] {n, sum};
  }
  /**
   * Indicates support of positive/negative approach, which can add and
   * remove values from the internal state; aggregate analytic functions
   * supporting that approach allow for a more efficient evaluation
   * when windows are used.
   * @return indicates whether PN approach is supported
   */
  @Override
  public boolean supportsPN() {
    return true;
  }
  /**
   * Indicates whether the internal state is empty.
   * @param state the internal state
   * @return indicates whether the internal state is empty
   */
  @Override
  public boolean isEmptyAggregate(Number[] state) {
    if ((Long) state[0] == 0)
      return true;
    return false;
  }
}
```

This example illustrates that the `state` object can track multiple properties. In this case `state` tracks both the count of the number of rows processed so far and the subtotal of the values for the column being averaged.

Kurtosis Using a Third Party Library

Kurtosis is a statistical measure of 'peakedness' in the values for a dataset compared to a normal distribution. This indicates how closely the distribution matches the rounded bell shape of a normal distribution.

In this example, we will use an implementation of kurtosis provided in the Apache Commons Math library, version 2.2. The method to calculate kurtosis, in the `DescriptiveStatistics` class in the Apache Library, expects the values to use as the probability distribution to be primitive values in an array.

To support this, the `aggregate` method builds an array from the column values for records in a group, partition or window. The `getAggregate` method then uses this array to perform the calculation. As always, the `state` object is used to hold state for both methods.

```
package com.raql.samples;
 import java.io.Serializable;
 import java.util.Arrays;
import org.apache.commons.math.stat.descriptive.DescriptiveStatistics;
 import com.jackbe.jbp.common.MessageSourceAccessor;
 import com.jackbe.jbp.raql.udx.loader.RaqlFunc;
 import de.rtm.push.adapters.DefaultSignature;
 import de.rtm.push.adapters.UserDefinedAggregationFunctionAdapter;
 import de.rtm.util.exception.UserDefinedAggregationFunctionException;
 import de.rtm.util.records.metadata.FieldMetaData;
 import de.rtm.util.records.metadata.FieldMetaDatas;
 import de.rtm.util.records.types.JavaTypes;
 import de.rtm.util.records.types.JavaTypes.Type;
@RaqlFunc(name="myKurtosisFunction")
/**
 * This adapter computes the kurtosis for a group, partition or window.
 */
public static class KurtosisAdapter implements UserDefinedAggregationFunctionAdapter<double[]> {
  /**
   * Returns the types of the input parameters.
   */
  @Override
  public Type[] getParameterTypes() {
    return new Type[] {Type.DOUBLE};
  }
  /**
   * Returns the type of the result.
   */
  @Override
  public Type getAggregateType() {
    return Type.DOUBLE;
  }
  /**
   * Creates the initial state of the kurtosis. Note that
   * the state solely stores the input values. Therefore
   * the initial state is an empty array.
   * @return the initial state of the kurtosis
   */
  @Override
  public double[] createInitialState() {
    return new double[] {};
  }
  /**
   * Aggregates the current internal state with the new input values
   * and derives a new internal state. As the kurtosis aggregate is
   * computed by an external library, the state solely stores
   * the incoming values.
   * @param state the current internal state
   * @param values the new input values
   * @return the new internal state
   */
  @Override
  public double[] aggregate(double[] state, Serializable[] values) {
    if (values[0] != null) {
      double[] newState = new double[state.length+1];
      System.arraycopy(state, 0, newState, 0, state.length);
      newState[newState.length-1] = (Double) values[0];
      return newState;
    }
    else
      return state;
  }
  /**
```

```
   * Derives the return value from the current internal state.
   * The kurtosis aggregate is computed by calling an external
   * library with all input values.
   * @param state the internal state
   * @return the kurtosis value
   */
  @Override
  public Object getAggregate(double[] state) {
    DescriptiveStatistics ds = new DescriptiveStatistics(state);
    double kurtosis = ds.getKurtosis();
    if (Double.isNaN(kurtosis))
      return 0.0;
    else
      return kurtosis;
  }
  /**
   * Removes the input values from the internal state.
   * This step is required for the support of the
   * positive/negative approach.
   * @param state the current internal state
   * @param values the input values
   * @return the new internal state
   */
  @Override
  public double[] negativeCall(double[] state, Serializable[] values) {
    if (values[0] == null)
      return state;
    else {
      boolean gotValue = false;
      double[] newState = new double[state.length-1];
      double value = (Double)values[0];
      for (int i = 0; i < state.length; i++) {
        gotValue |= state[i] == value;
        newState[i] = state[i + (gotValue ? 1 :0)];
      }
      return newState;
    }
  }
  /**
   * Indicates support for positive/negative approach, which allows
   * adding or removing values from the internal state;
   * This approach provides a more efficient evaluation
   * when windows are used.
   * @return indicates whether PN approach is supported
   */
  @Override
  public boolean supportsPN() {
    return true;
  }
  /**
   * Indicates whether the internal state is empty.
   * @param state the internal state
   * @return indicates whether the internal state is empty
   */
  @Override
  public boolean isEmptyAggregate(double[] state) {
    return state.length == 0;
  }
}
```

To compile this example, you must include the Apache Commons Math library, version 2.2, in the classpath. You may include the jar file for this library in the lib folder for the user-defined function library. This specific library is also used in MashZone NextGen,

so you also simply add the jar file for this library to the classpath. See "Statistics and Analytics Third-Party Libraries" on page 823 for information.

Use a Factory for Function Overloading

If a user-defined function must support multiple signatures, you must implement the corresponding factory interface:

- `UserDefinedFunctionAdapterFactory` for plain functions
- `UserDefinedAggregateFunctionAdapterFactory` for aggregate analytic functions
- `UserDefinedWindowFunctionAdapterFactory` for window analytic functions

For more information on the advantages of user-defined function factories and examples. See the RAQL User-Defined Function API reference.

Statistics and Analytics Third-Party Libraries

MashZone NextGen Analytics includes the following third-party libraries with many common statistics and machine learning algorithms:

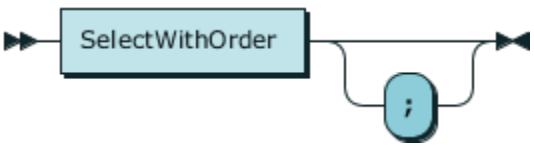
- Apache Commons Mathematics Library, version 2.2. Include *web-apps-home* /presto/WEB-INF/lib/commons-math-2.2.jar in the classpath.
- Apache Mahout Library, version 0.7. Include these jar files in the class path:
  - *web-apps-home* /presto/WEB-INF/lib/hadoop-core-1.1.1.jar
  - *web-apps-home* /presto/WEB-INF/lib/mahout-core-0.7.jar
  - *web-apps-home* /presto/WEB-INF/lib/mahout-core-0.7-job.jar
  - *web-apps-home* /presto/WEB-INF/lib/mahout-integration-0.7.jar
  - *web-apps-home* /presto/WEB-INF/lib/mahout-math-0.7.jar

User-defined analytics functions can leverage these libraries directly. Simply include the associated jar file in the classpath along with the jar for RAQL when you compile your UDF classes.

RAQL Query Syntax Reference

This topic contains syntax definitions and corresponding syntax diagrams for the Real-Time Analytics Query Language (RAQL) in MashZone NextGen.

SelectStatement



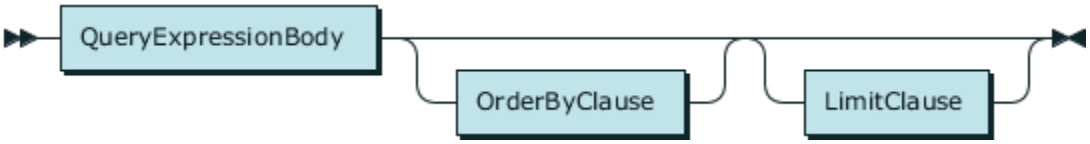
Definition:

```
SelectStatement
        ::= SelectWithOrder ';'?
```

Used In:

This is the starting definition of RAQL queries, which has no parent references.

SelectWithOrder



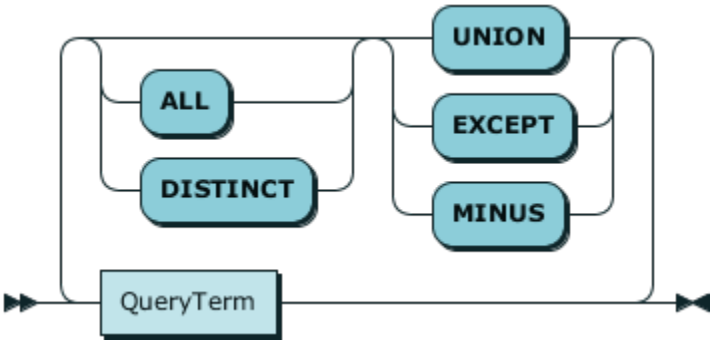
Definition:

```
SelectWithOrder
        ::= QueryExpressionBody OrderByClause? LimitClause?
```

Used In:

- SelectStatement
- SourceReferenceOrSubquery

QueryExpressionBody



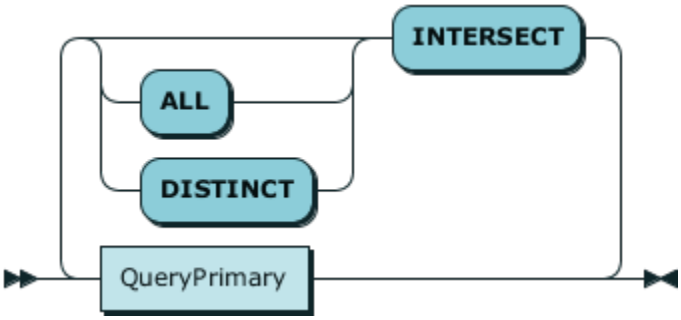
Definition:

```
QueryExpressionBody
        ::= QueryTerm ( ( 'UNION' | 'EXCEPT' | 'MINUS' ) ( 'ALL' | 'DISTINCT' )? QueryTerm )*
```

Used In:

- QueryPrimary
- SelectWithOrder
- SqlRelationalOperatorExpression

QueryTerm



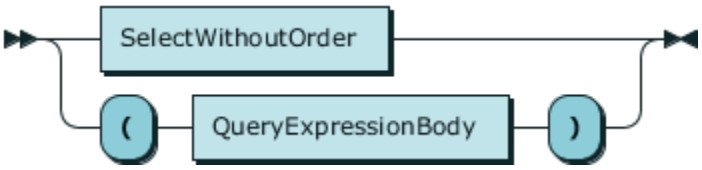
Definition:

```
QueryTerm
        ::= QueryPrimary ( 'INTERSECT' ( 'ALL' | 'DISTINCT' )? QueryPrimary )*
```

Used In:

- QueryExpressionBody

QueryPrimary



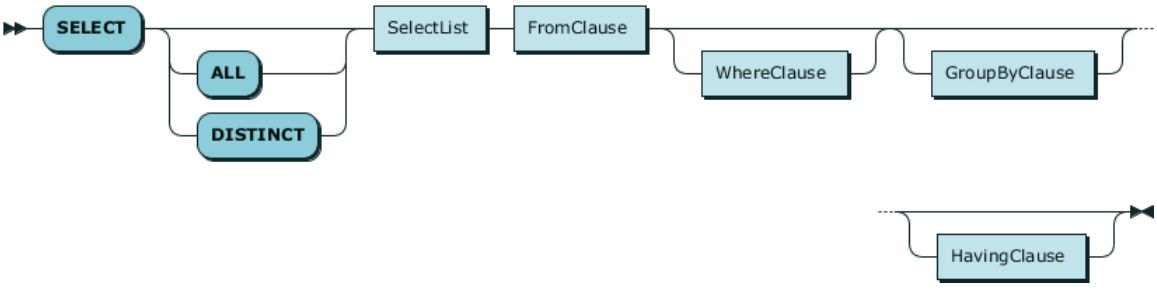
Definition:

```
QueryPrimary
        ::= SelectWithoutOrder
          | '(' QueryExpressionBody ')'
```

Used In:

- QueryTerm

SelectWithoutOrder



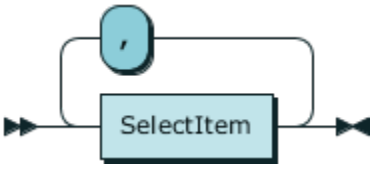
Definition:

```
SelectWithoutOrder
        ::= 'SELECT' ( 'ALL' | 'DISTINCT' )? SelectList FromClause WhereClause? GroupByClause? H
```

Used In:

- QueryPrimary

SelectList



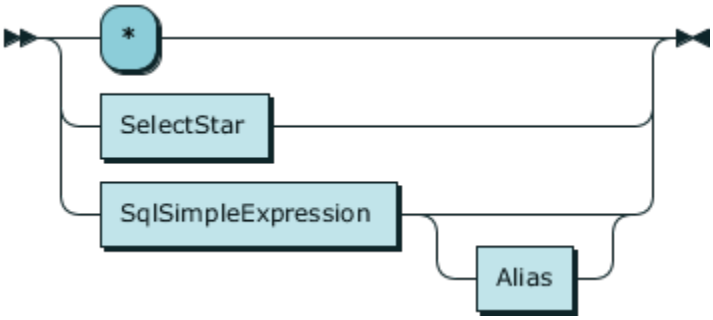
Definition:

```
SelectList
        ::= SelectItem ( ',' SelectItem )*
```

Used In:

- SelectWithoutOrder

SelectItem



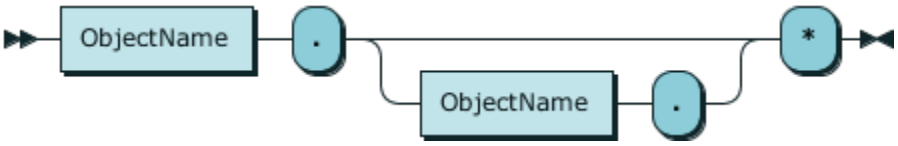
Definition:

```
SelectItem
        ::= '*'
          | SelectStar
          | SqlSimpleExpression Alias?
```

Used In:

- SelectList

SelectStar



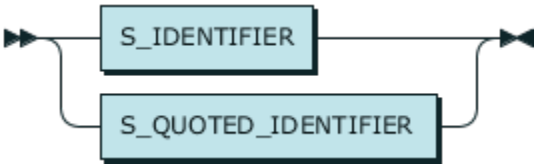
Definition:

```
SelectStar
        ::= ObjectName '.' ( ObjectName '.' )? '*'
```

Used In:

- SelectItem

ObjectName



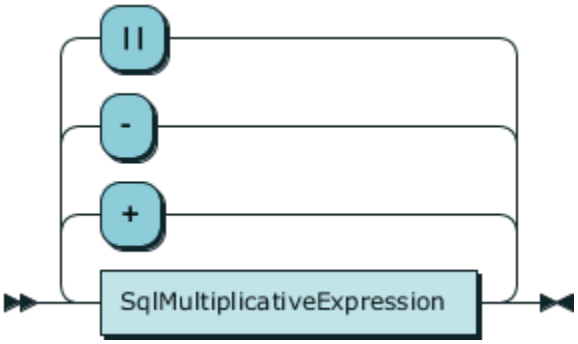
Definition:

```
ObjectName
         ::= S_IDENTIFIER
           | S_QUOTED_IDENTIFIER
```

Used In:

- Alias
- GenericFunctionCall
- SelectStar
- SourceField
- SourceReference
- SqlPrimaryExpression

SqlSimpleExpression



Definition:

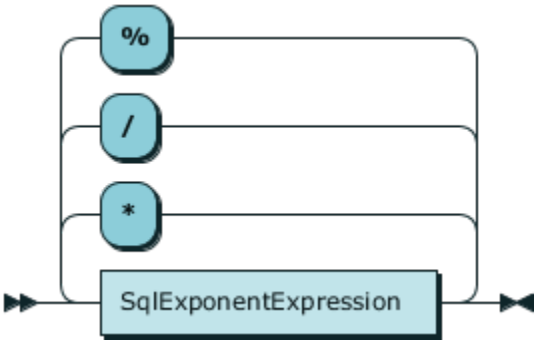
```
SqlSimpleExpression
         ::= SqlMultiplicativeExpression ( ( '+' | '-' | '||' ) SqlMultiplicativeExpression )*
```

Used In:

- BooleanCaseCall
- SelectItem
- SimpleCaseCall
- SqlBetweenClause
- SqlLikeClause

- SqlRelationalExpression
- SqlRelationalOperatorExpression

SqlMultiplicativeExpression



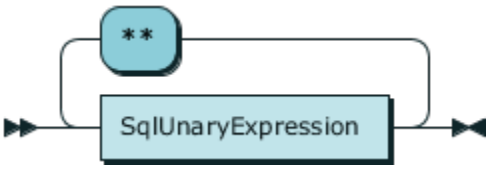
Definition:

```
SqlMultiplicativeExpression
        ::= SqlExponentExpression ( ( '*' | '/' | '%' ) SqlExponentExpression )*
```

Used In:

- SqlSimpleExpression

SqlExponentExpression



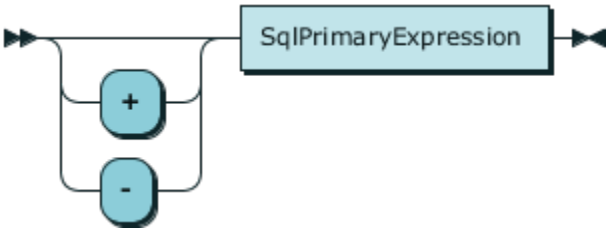
Definition:

```
SqlExponentExpression
        ::= SqlUnaryExpression ( '**' SqlUnaryExpression )*
```

Used In:

- SqlMultiplicativeExpression

SqlUnaryExpression



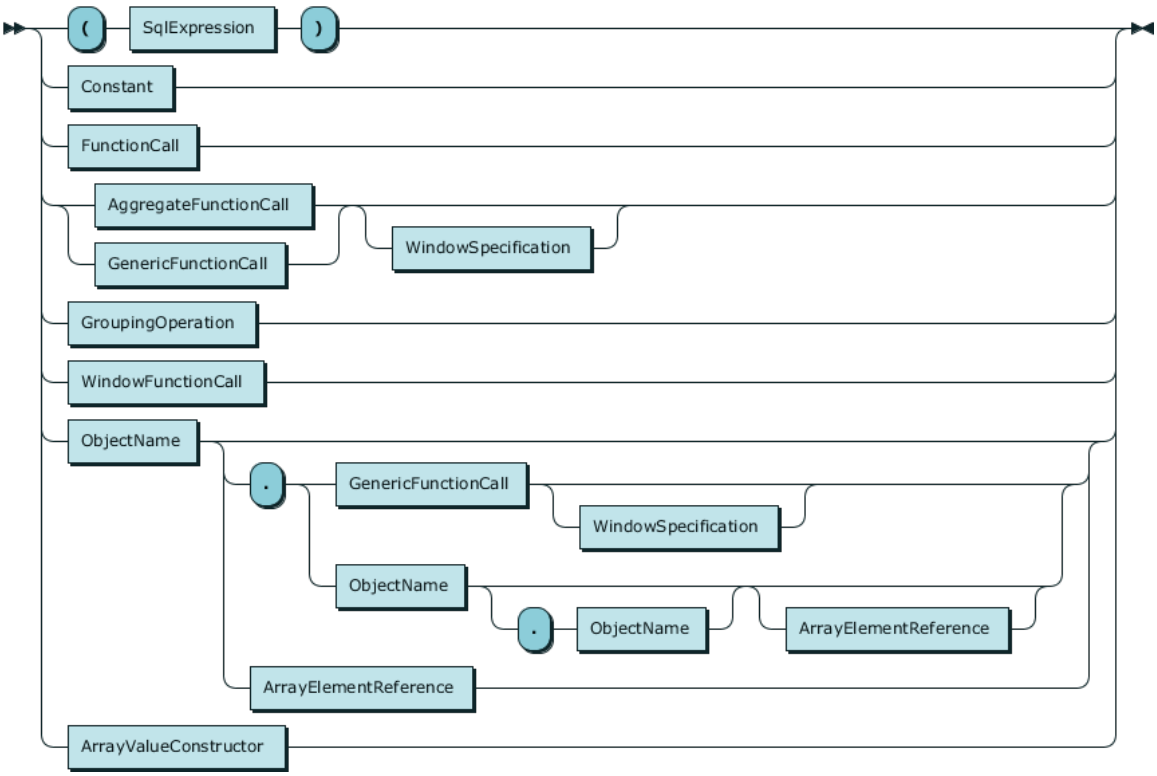
Definition:

```
SqlUnaryExpression
        ::= ( '+' | '-' )? SqlPrimaryExpression
```

Used In:

- SqlExponentExpression

SqlPrimaryExpression



Definition:

```
SqlPrimaryExpression
         ::= '(' SqlExpression ')'
           | Constant
           | FunctionCall
           | ( AggregateFunctionCall | GenericFunctionCall ) WindowSpecification?
           | GroupingOperation
           | WindowFunctionCall
           | ObjectName ( '.' ( GenericFunctionCall WindowSpecification? | ObjectName ( '.' Objec
           | ArrayValueConstructor
```

Used In:

- SqlUnaryExpression

SqlExpression

Definition:

```
SqlExpression
        ::= SqlAndExpression ( 'OR' SqlAndExpression )*
```

Used In:

- AggregateFunctionCall
- BooleanCaseCall
- FunctionCall
- GroupingExpressionReference
- HavingClause
- JoinedTable
- OrderByClause
- SqlExpressionList
- SqlPrimaryExpression
- WhereClause
- WindowFunctionCall

SqlAndExpression

Definition:

```
SqlAndExpression
        ::= SqlUnaryLogicalExpression ( 'AND' SqlUnaryLogicalExpression )*
```

Used In:

- SqlExpression

SqlUnaryLogicalExpression

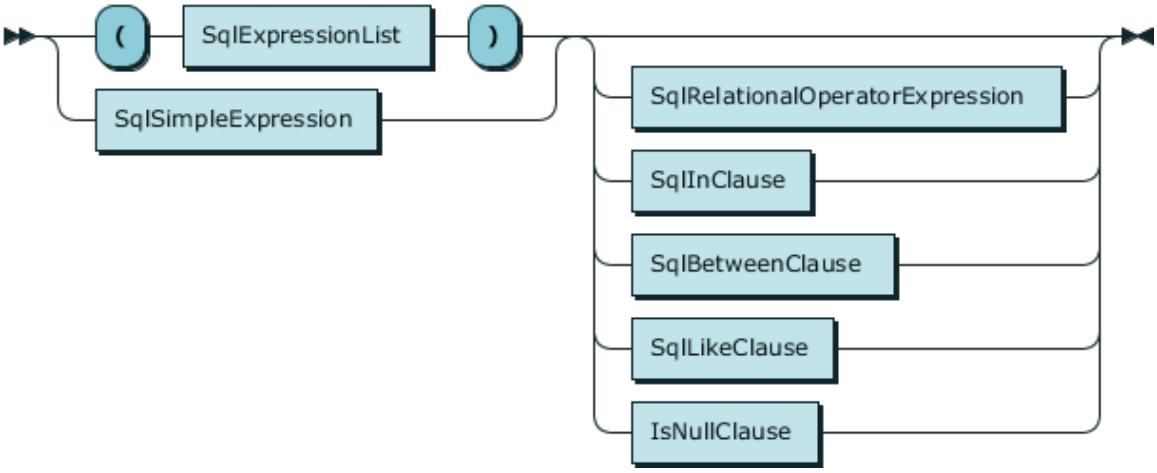
Definition:

```
SqlUnaryLogicalExpression
        ::= 'NOT'? SqlRelationalExpression
```

Used In:

- SqlAndExpression

SqlRelationalExpression



Definition:

```
SqlRelationalExpression
        ::= ( '(' SqlExpressionList ')' | SqlSimpleExpression ) ( SqlRelationalOperatorExpressio
```

Used In:

- SqlUnaryLogicalExpression

SqlExpressionList

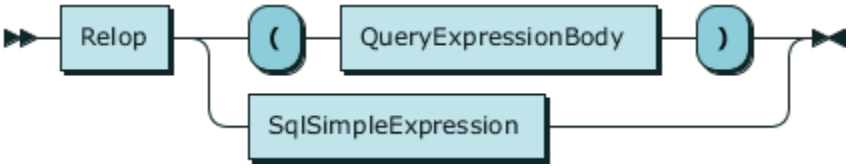
Definition:

```
SqlExpressionList
        ::= SqlExpression ( ',' SqlExpression )*
```

Used In:

- ArrayValueConstructor
- GenericFunctionCall
- GroupingOperation
- SqlInClause
- SqlRelationalExpression
- WindowSpecification

SqlRelationalOperatorExpression



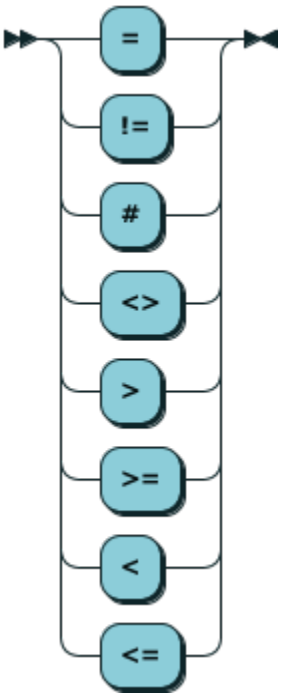
Definition:

```
SqlRelationalOperatorExpression
        ::= Relop ( '(' QueryExpressionBody ')' | SqlSimpleExpression )
```

Used In:

- SqlRelationalExpression

Relop



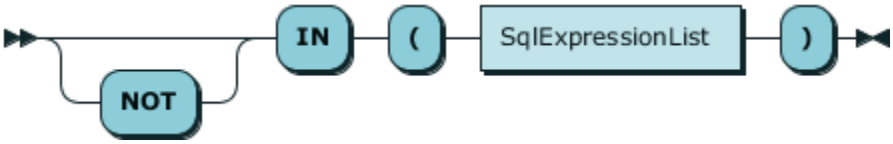
Definition:

```
Relop    ::= '='
           | '!='
           | '#'
           | '<>'
           | '>'
           | '>='
           | '<'
           | '<='
```

Used In:

- SqlRelationalOperatorExpression

SqlInClause



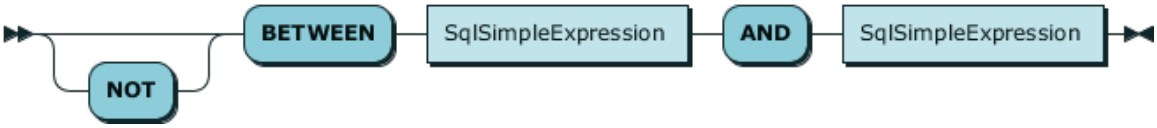
Definition:

```
SqlInClause
        ::= 'NOT'? 'IN' '(' SqlExpressionList ')'
```

Used In:

- SqlRelationalExpression

SqlBetweenClause



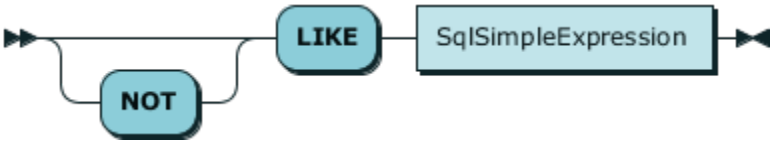
Definition:

```
SqlBetweenClause
        ::= 'NOT'? 'BETWEEN' SqlSimpleExpression 'AND' SqlSimpleExpression
```

Used In:

- SqlRelationalExpression

SqlLikeClause



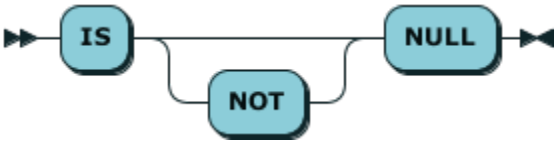
Definition:

```
SqlLikeClause
        ::= 'NOT'? 'LIKE' SqlSimpleExpression
```

Used In:

- SqlRelationalExpression

IsNullClause



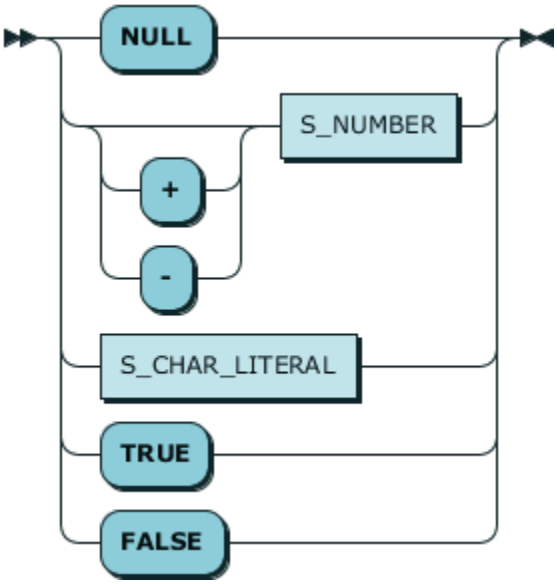
Definition:

```
IsNullClause
        ::= 'IS' 'NOT'? 'NULL'
```

Used In:

- SqlRelationalExpression

Constant



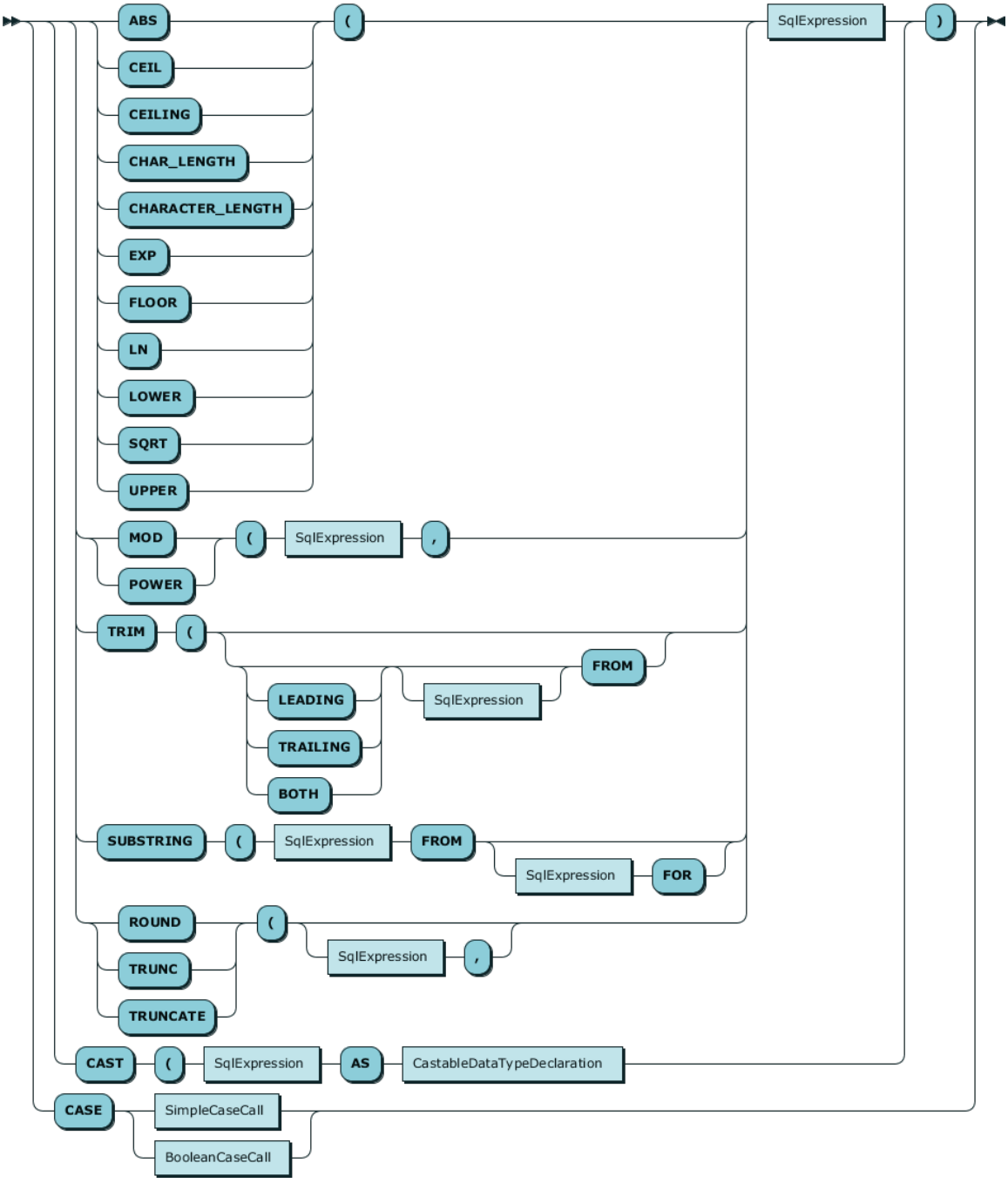
Definition:

```
Constant ::= 'NULL'
           | ( '+' | '-' )? S_NUMBER
           | S_CHAR_LITERAL
           | 'TRUE'
           | 'FALSE'
```

Used In:

- ConstantList
- SqlPrimaryExpression
- WindowFunctionCall

FunctionCall



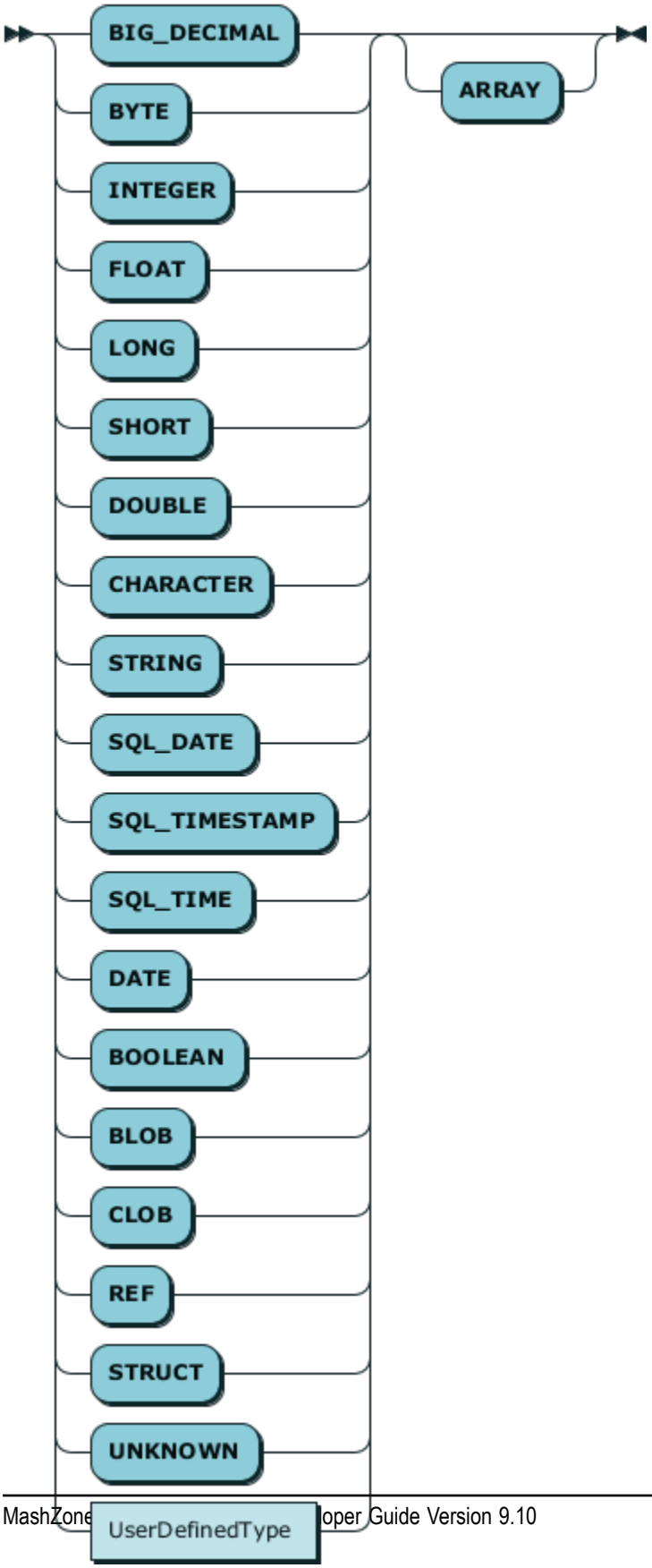
Definition:

```
FunctionCall
        ::= ( ( ( 'ABS' | 'CEIL' | 'CEILING' | 'CHAR_LENGTH' | 'CHARACTER_LENGTH' | 'EXP' | 'FLO
          | 'CASE' ( SimpleCaseCall | BooleanCaseCall )
```

Used In:

- SqlPrimaryExpression

CastableDataTypeDeclaration



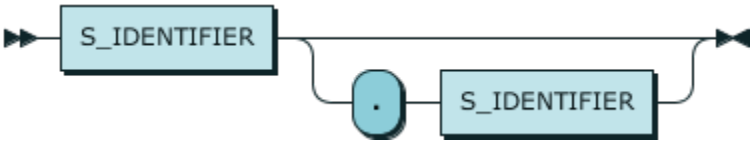
Definition:

```
CastableDataTypeDeclaration
        ::= ( 'BIG_DECIMAL' | 'BYTE' | 'INTEGER' | 'FLOAT' | 'LONG' | 'SHORT' | 'DOUBLE' | 'CHAR
```

Used In:

- FunctionCall

UserDefinedType



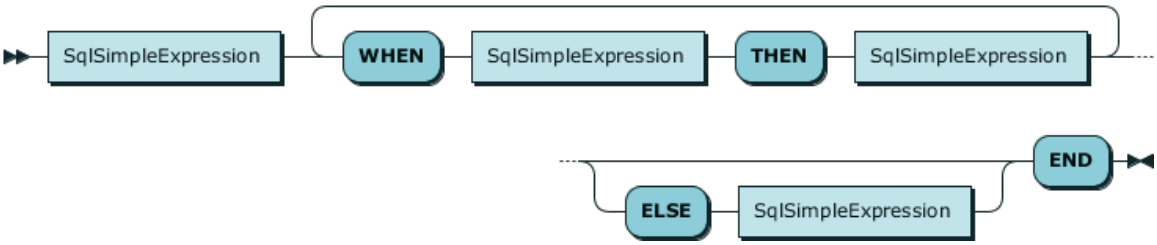
Definition:

```
UserDefinedType
        ::= S_IDENTIFIER ( '.' S_IDENTIFIER )?
```

Used In:

- CastableDataTypeDeclaration

SimpleCaseCall



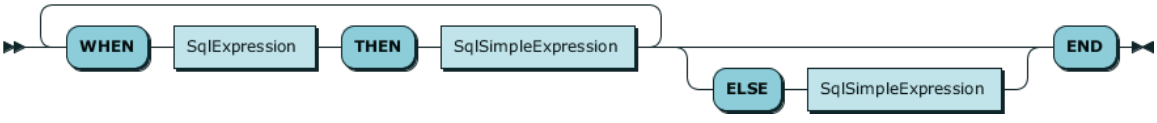
Definition:

```
SimpleCaseCall
        ::= SqlSimpleExpression ( 'WHEN' SqlSimpleExpression 'THEN' SqlSimpleExpression )+ ( 'EL
```

Used In:

- FunctionCall

BooleanCaseCall



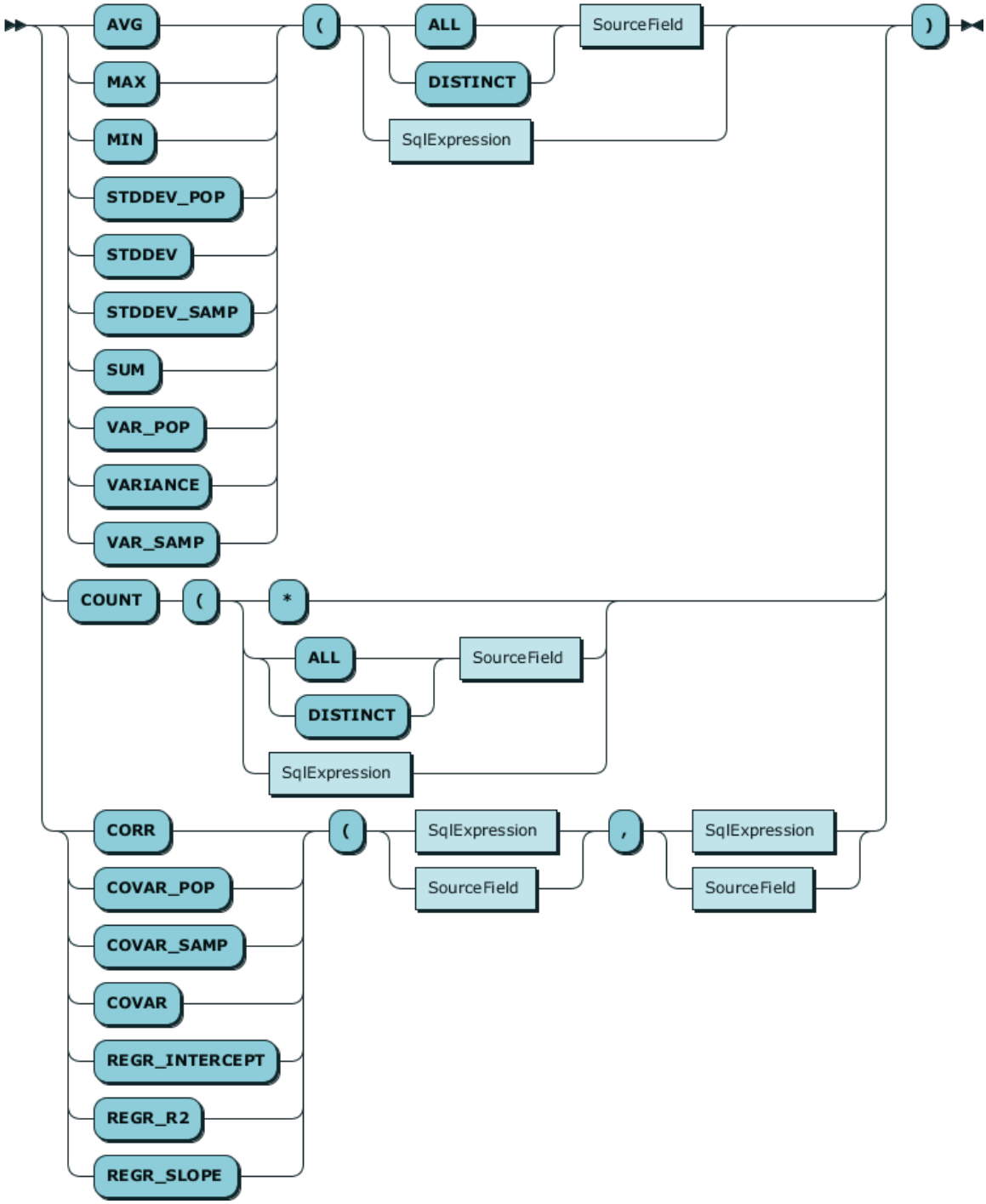
Definition:

```
BooleanCaseCall
        ::= ( 'WHEN' SqlExpression 'THEN' SqlSimpleExpression )+ ( 'ELSE' SqlSimpleExpression )?
```

Used In:

- FunctionCall

AggregateFunctionCall



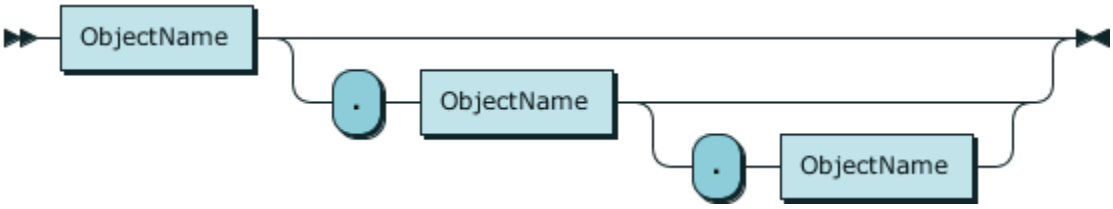
Definition:

```
AggregateFunctionCall
        ::= ( ( 'AVG' | 'MAX' | 'MIN' | 'STDDEV_POP' | 'STDDEV' | 'STDDEV_SAMP' | 'SUM' | 'VAR_P
```

Used In:

- SqlPrimaryExpression

SourceField



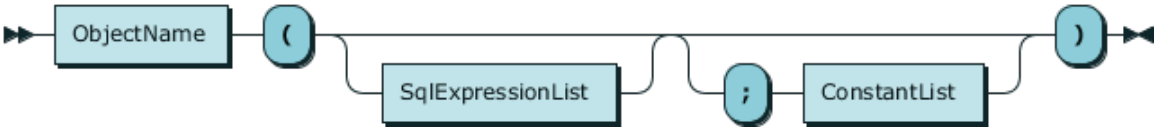
Definition:

```
SourceField
        ::= ObjectName ( '.' ObjectName ( '.' ObjectName )? )?
```

Used In:

- AggregateFunctionCall
- SourceFieldList
- WindowFunctionCall

GenericFunctionCall



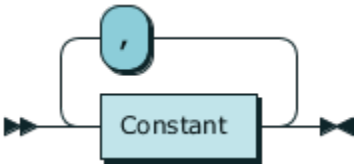
Definition:

```
GenericFunctionCall
        ::= ObjectName '(' SqlExpressionList? ( ';' ConstantList )? ')'
```

Used In:

- SqlPrimaryExpression

ConstantList



Definition:

```
ConstantList
        ::= Constant ( ',' Constant )*
```

Used In:

- GenericFunctionCall

WindowSpecification

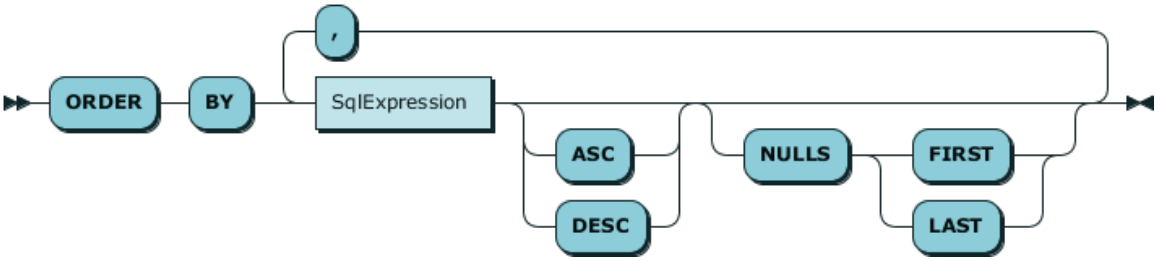
Definition:

```
WindowSpecification
        ::= 'OVER' '(' ( 'PARTITION' 'BY' SqlExpressionList )? OrderByClause? WindowFrameClause?
```

Used In:

- SqlPrimaryExpression
- WindowFunctionCall

OrderByClause



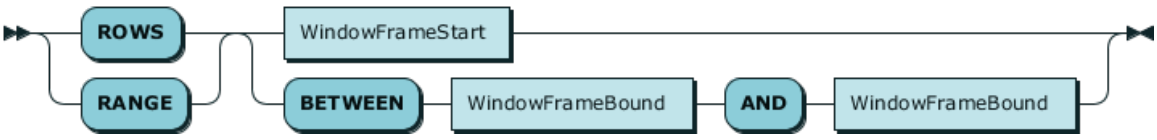
Definition:

```
OrderByClause
        ::= 'ORDER' 'BY' SqlExpression ( 'ASC' | 'DESC' )? ( 'NULLS' ( 'FIRST' | 'LAST' ) )? ( '
```

Used In:

- SelectWithOrder
- WindowSpecification

WindowFrameClause



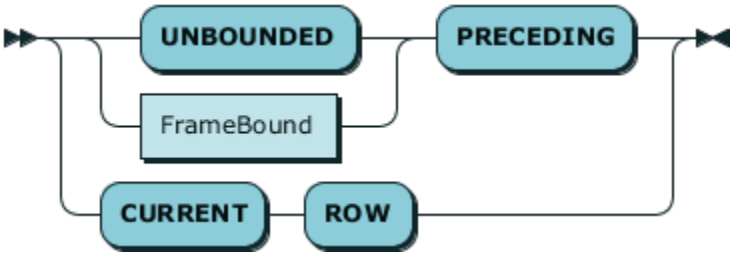
Definition:

```
WindowFrameClause
        ::= ( 'ROWS' | 'RANGE' ) ( WindowFrameStart | 'BETWEEN' WindowFrameBound 'AND' WindowFra
```

Used In:

- WindowSpecification

WindowFrameStart



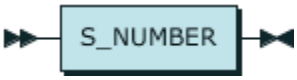
Definition:

```
WindowFrameStart
         ::= ( 'UNBOUNDED' | FrameBound ) 'PRECEDING'
           | 'CURRENT' 'ROW'
```

Used In:

- WindowFrameBound
- WindowFrameClause

FrameBound



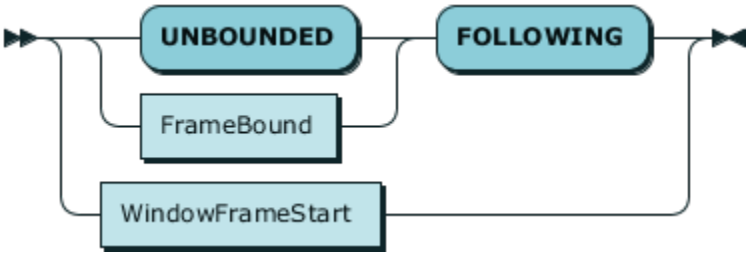
Definition:

```
FrameBound
         ::= S_NUMBER
```

Used In:

- WindowFrameBound
- WindowFrameStart

WindowFrameBound



Definition:

```
WindowFrameBound
         ::= ( 'UNBOUNDED' | FrameBound ) 'FOLLOWING'
           | WindowFrameStart
```

Used In:

- WindowFrameClause

GroupingOperation

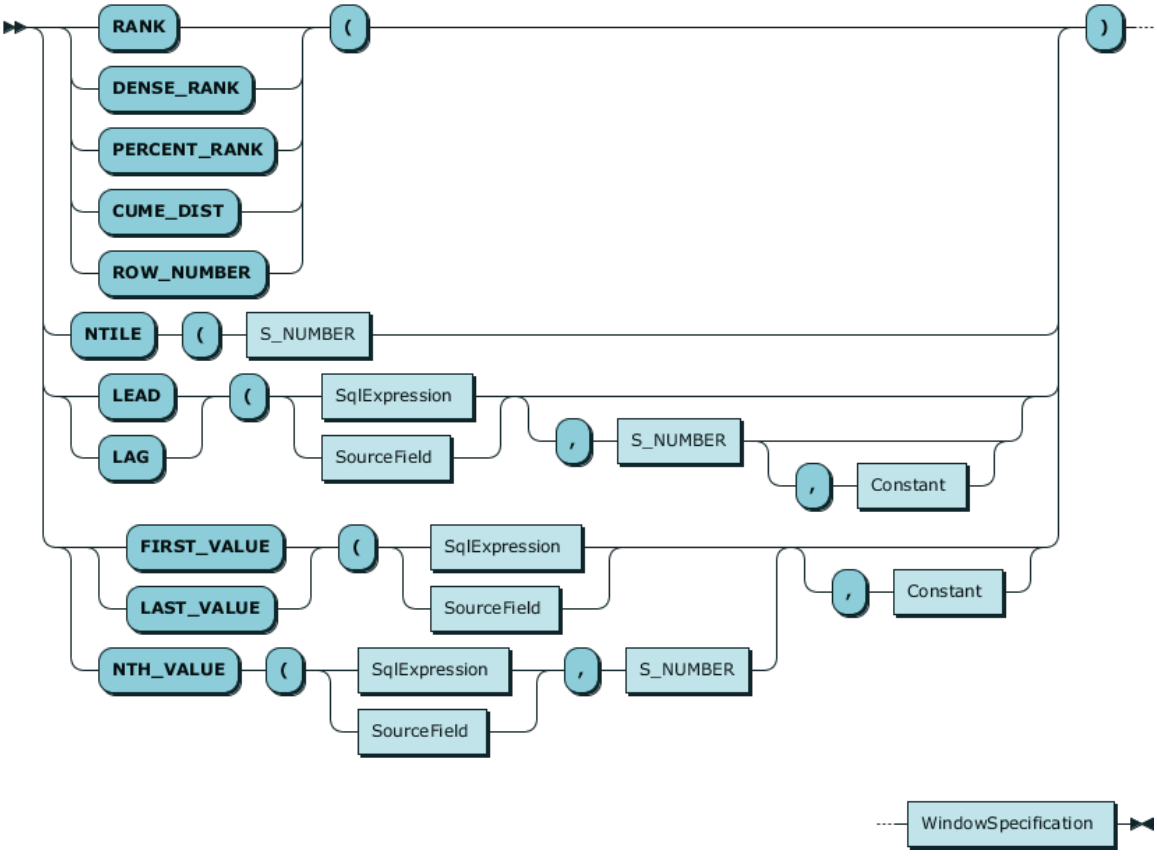
Definition:

```
GroupingOperation
        ::= 'GROUPING' '(' SqlExpressionList ')'
```

Used In:

- SqlPrimaryExpression

WindowFunctionCall



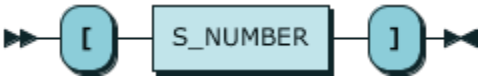
Definition:

```
WindowFunctionCall
        ::= ( ( 'RANK' | 'DENSE_RANK' | 'PERCENT_RANK' | 'CUME_DIST' | 'ROW_NUMBER' ) '(' | 'NTI
```

Used In:

- SqlPrimaryExpression

ArrayElementReference



Definition:

```
ArrayElementReference
        ::= '[' S_NUMBER ']'
```

Used In:

- SqlPrimaryExpression

ArrayValueConstructor

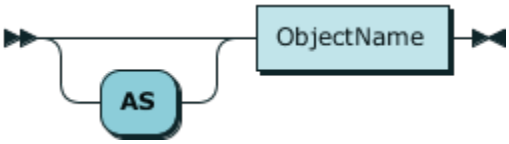
Definition:

```
ArrayValueConstructor
        ::= 'ARRAY' '[' SqlExpressionList ']'
```

Used In:

- SqlPrimaryExpression

Alias



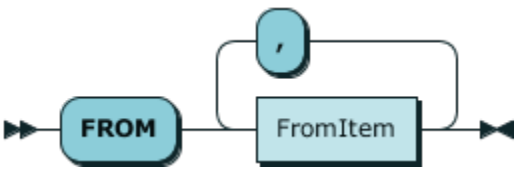
Definition:

```
Alias    ::= 'AS'? ObjectName
```

Used In:

- SelectItem
- SourceReferenceOrSubquery

FromClause



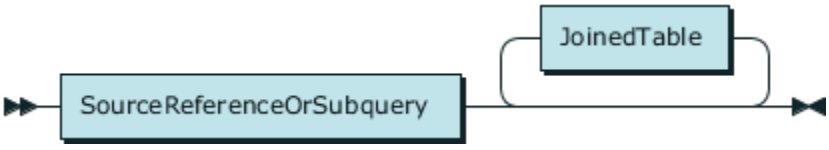
Definition:

```
FromClause
        ::= 'FROM' FromItem ( ',' FromItem )*
```

Used In:

- SelectWithoutOrder

FromItem



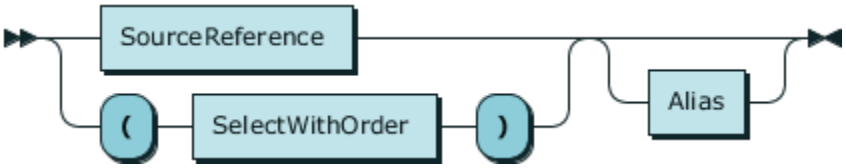
Definition:

```
FromItem ::= SourceReferenceOrSubquery JoinedTable*
```

Used In:

- FromClause

SourceReferenceOrSubquery



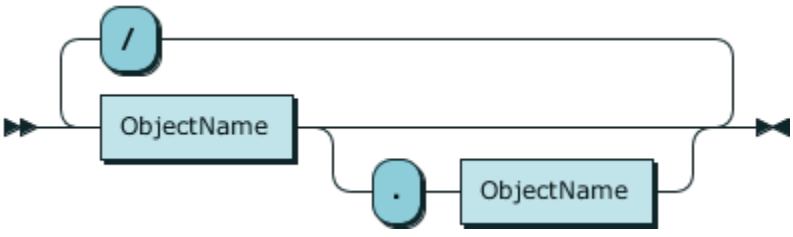
Definition:

```
SourceReferenceOrSubquery
        ::= ( SourceReference | '(' SelectWithOrder ')' ) Alias?
```

Used In:

- FromItem
- JoinedTable

SourceReference



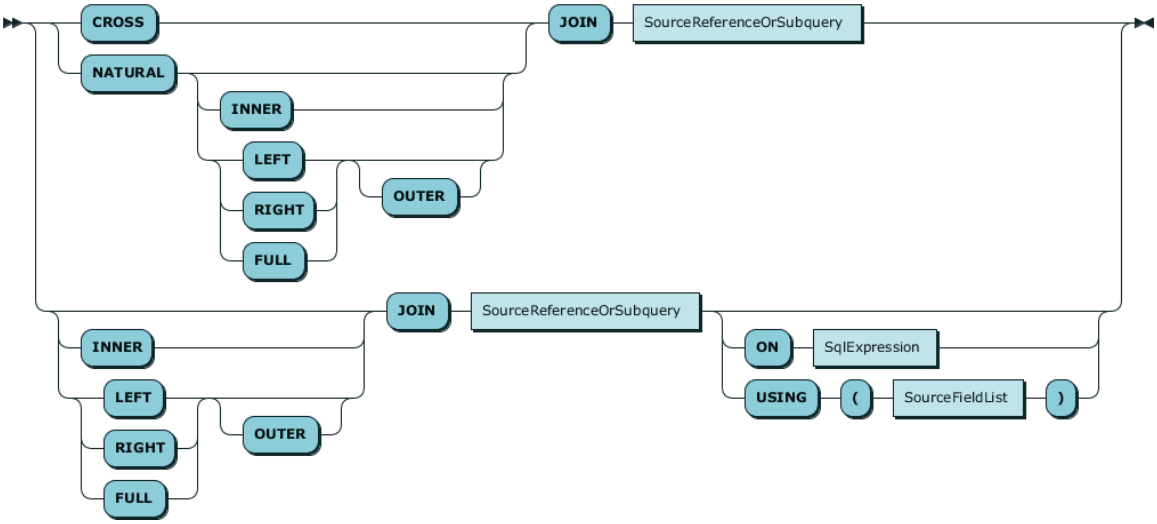
Definition:

```
SourceReference
        ::= ObjectName ( '.' ObjectName )? ( '/' ObjectName ( '.' ObjectName )? )*
```

Used In:

- SourceReferenceOrSubquery

JoinedTable



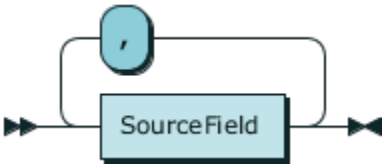
Definition:

```
JoinedTable
        ::= ( 'CROSS' | 'NATURAL' ( 'INNER' | ( 'LEFT' | 'RIGHT' | 'FULL' ) 'OUTER'? )? ) 'JOIN'
          | ( 'INNER' | ( 'LEFT' | 'RIGHT' | 'FULL' ) 'OUTER'? )? 'JOIN' SourceReferenceOrSubque
```

Used In:

- FromItem

SourceFieldList



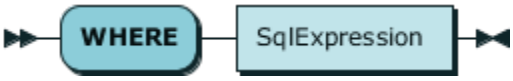
Definition:

```
SourceFieldList
        ::= SourceField ( ',' SourceField )*
```

Used In:

- JoinedTable

WhereClause



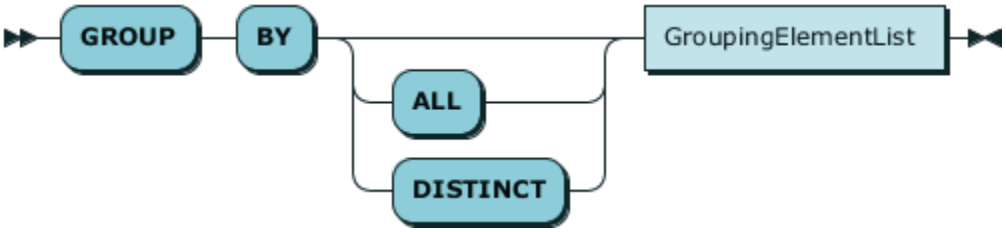
Definition:

```
WhereClause
        ::= 'WHERE' SqlExpression
```

Used In:

- SelectWithoutOrder

GroupByClause



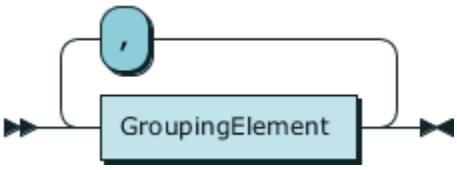
Definition:

```
GroupByClause
        ::= 'GROUP' 'BY' ( 'ALL' | 'DISTINCT' )? GroupingElementList
```

Used In:

- SelectWithoutOrder

GroupingElementList



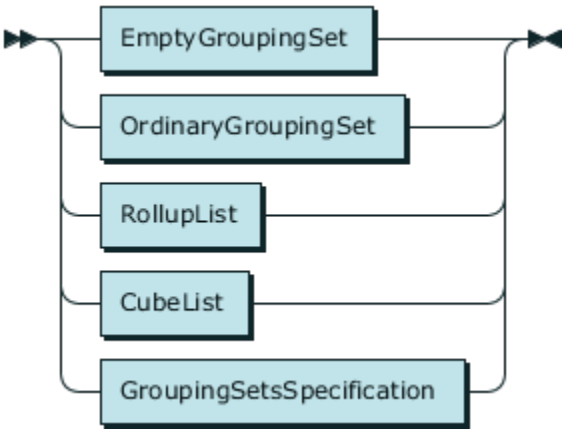
Definition:

```
GroupingElementList
        ::= GroupingElement ( ',' GroupingElement )*
```

Used In:

- GroupByClause

GroupingElement



Definition:

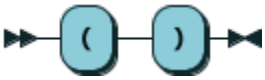
```
GroupingElement
```

```
          ::= EmptyGroupingSet
            | OrdinaryGroupingSet
            | RollupList
            | CubeList
            | GroupingSetsSpecification
```

Used In:

- GroupingElementList

EmptyGroupingSet



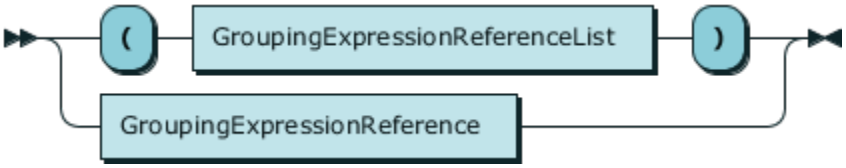
Definition:

```
EmptyGroupingSet
          ::= '(' ')'
```

Used In:

- GroupingElement
- GroupingSet

OrdinaryGroupingSet



Definition:

```
OrdinaryGroupingSet
          ::= '(' GroupingExpressionReferenceList ')'
            | GroupingExpressionReference
```

Used In:

- GroupingElement
- GroupingSet
- OrdinaryGroupingSetList

GroupingExpressionReferenceList

Definition:

```
GroupingExpressionReferenceList
          ::= GroupingExpressionReference ( ',' GroupingExpressionReference )*
```

Used In:

- OrdinaryGroupingSet

GroupingExpressionReference

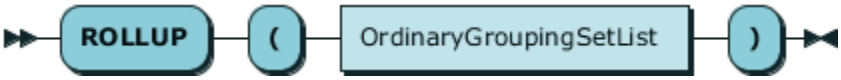
Definition:

```
GroupingExpressionReference
        ::= SqlExpression
```

Used In:

- GroupingExpressionReferenceList
- OrdinaryGroupingSet

RollupList



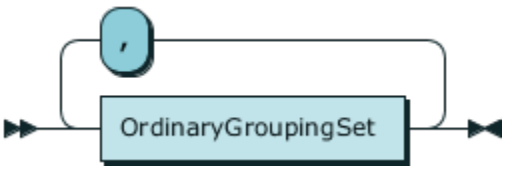
Definition:

```
RollupList
        ::= 'ROLLUP' '(' OrdinaryGroupingSetList ')'
```

Used In:

- GroupingElement
- GroupingSet

OrdinaryGroupingSetList



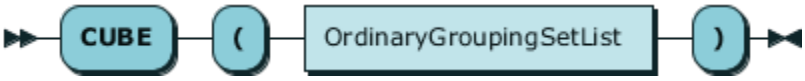
Definition:

```
OrdinaryGroupingSetList
        ::= OrdinaryGroupingSet ( ',' OrdinaryGroupingSet )*
```

Used In:

- CubeList
- RollupList

CubeList



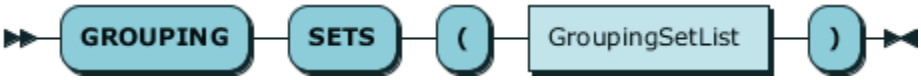
Definition:

```
CubeList ::= 'CUBE' '(' OrdinaryGroupingSetList ')'
```

Used In:

- GroupingElement
- GroupingSet

GroupingSetsSpecification



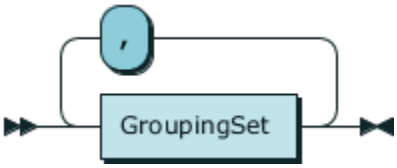
Definition:

```
GroupingSetsSpecification
         ::= 'GROUPING' 'SETS' '(' GroupingSetList ')'
```

Used In:

- GroupingElement
- GroupingSet

GroupingSetList



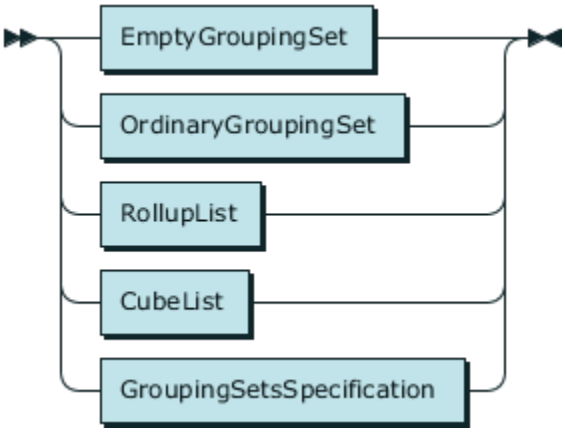
Definition:

```
GroupingSetList
         ::= GroupingSet ( ',' GroupingSet )*
```

Used In:

- GroupingSetsSpecification

GroupingSet



Definition:

```
GroupingSet
         ::= EmptyGroupingSet
           | OrdinaryGroupingSet
           | RollupList
           | CubeList
           | GroupingSetsSpecification
```

Used In:

- GroupingSetList

HavingClause

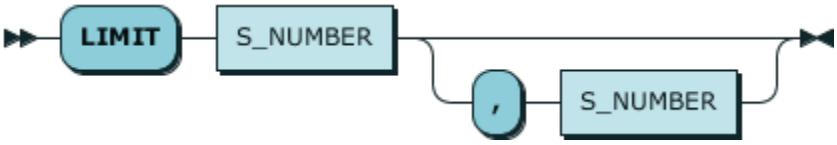
Definition:

```
HavingClause
         ::= 'HAVING' SqlExpression
```

Used In:

- SelectWithoutOrder

LimitClause



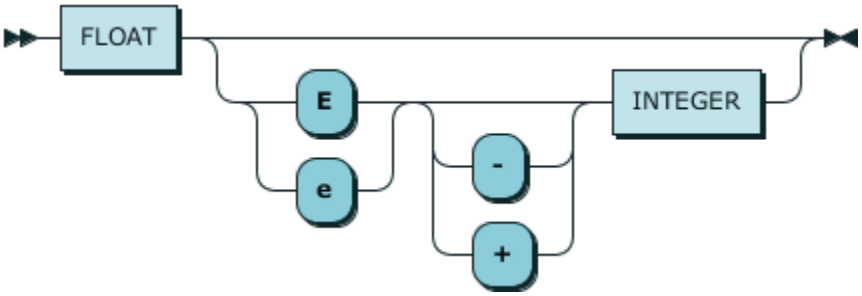
Definition:

```
LimitClause
         ::= 'LIMIT' S_NUMBER ( ',' S_NUMBER )?
```

Used In:

- SelectWithOrder

S_NUMBER



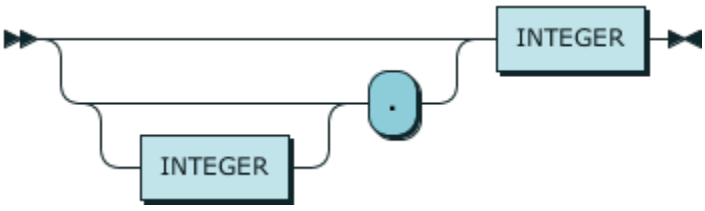
Definition:

```
S_NUMBER ::= FLOAT ( [Ee] [-+]? INTEGER )?
```

Used In:

- ArrayElementReference
- Constant
- FrameBound
- LimitClause
- WindowFunctionCall

FLOAT



Definition:

```
FLOAT    ::= ( INTEGER? '.' )? INTEGER
```

Used In:

- S_NUMBER

INTEGER

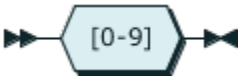
Definition:

```
INTEGER  ::= DIGIT+
```

Used In:

- FLOAT
- S_NUMBER

DIGIT



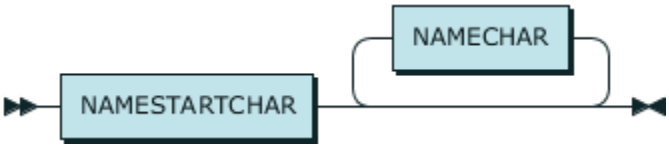
Definition:

```
DIGIT    ::= [0-9]
```

Used In:

- INTEGER

S_IDENTIFIER



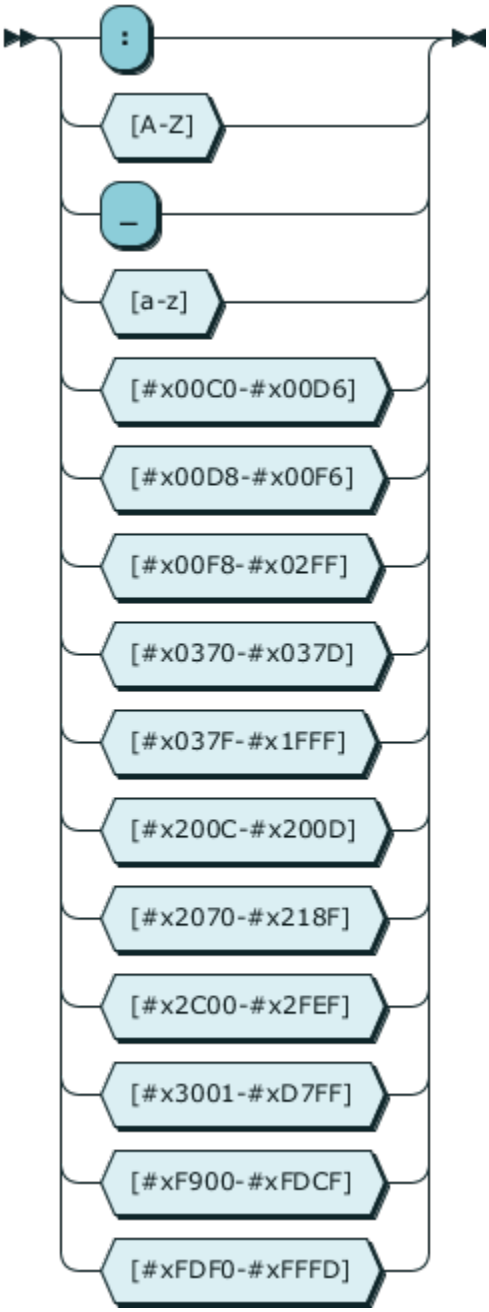
Definition:

```
S_IDENTIFIER
         ::= NAMESTARTCHAR NAMECHAR*
```

Used In:

- ObjectName
- UserDefinedType

NAMESTARTCHAR



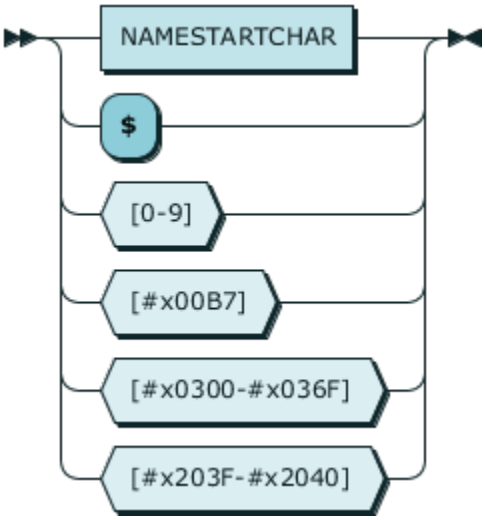
Definition:

```
NAMESTARTCHAR
         ::= [:A-Z_a-z#x00C0-#x00D6#x00D8-#x00F6#x00F8-#x02FF#x0370-#x037D#x037F-#x1FFF#x200C-#x2
```

Used In:

- NAMECHAR
- S_IDENTIFIER

NAMECHAR



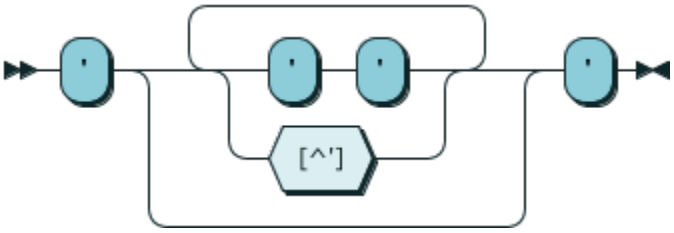
Definition:

```
NAMECHAR ::= NAMESTARTCHAR
           | [$0-9#x00B7#x0300-#x036F#x203F-#x2040]
```

Used In:

- S_IDENTIFIER

S_CHAR_LITERAL



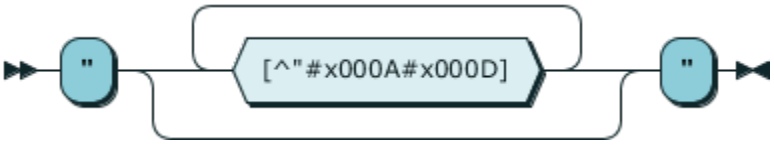
Definition:

```
S_CHAR_LITERAL
         ::= "'" ( "'" "'" | [^'] )* "'"
```

Used In:

- Constant

S_QUOTED_IDENTIFIER



Definition:

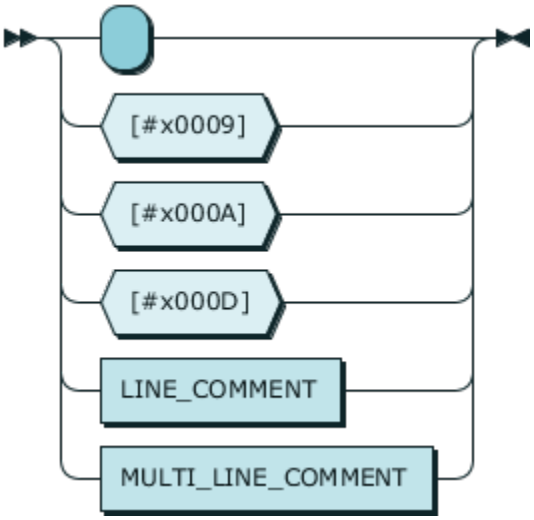
```
S_QUOTED_IDENTIFIER
```

```
        ::= '"' [^"#x000A#x000D]* '"'
```

Used In:

- ObjectName

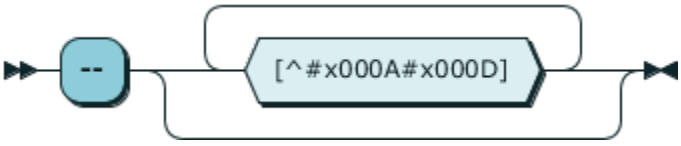
IgnorableWhitespace



Definition:

```
IgnorableWhitespace
        ::= [ #x0009#x000A#x000D]
          | LINE_COMMENT
          | MULTI_LINE_COMMENT
```

LINE_COMMENT



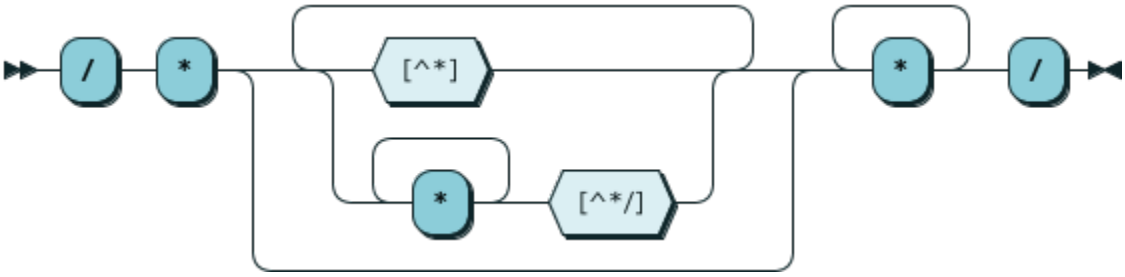
Definition:

```
LINE_COMMENT
        ::= '--' [^#x000A#x000D]*
```

Used In:

- IgnorableWhitespace

MULTI_LINE_COMMENT



Definition:

```
MULTI_LINE_COMMENT
        ::= '/' '*' ( [^*] | '*'+ [^*/] )* '*'+ '/'
```

Used In:

- IgnorableWhitespace

Working with MashZone NextGen Analytics In-Memory Stores

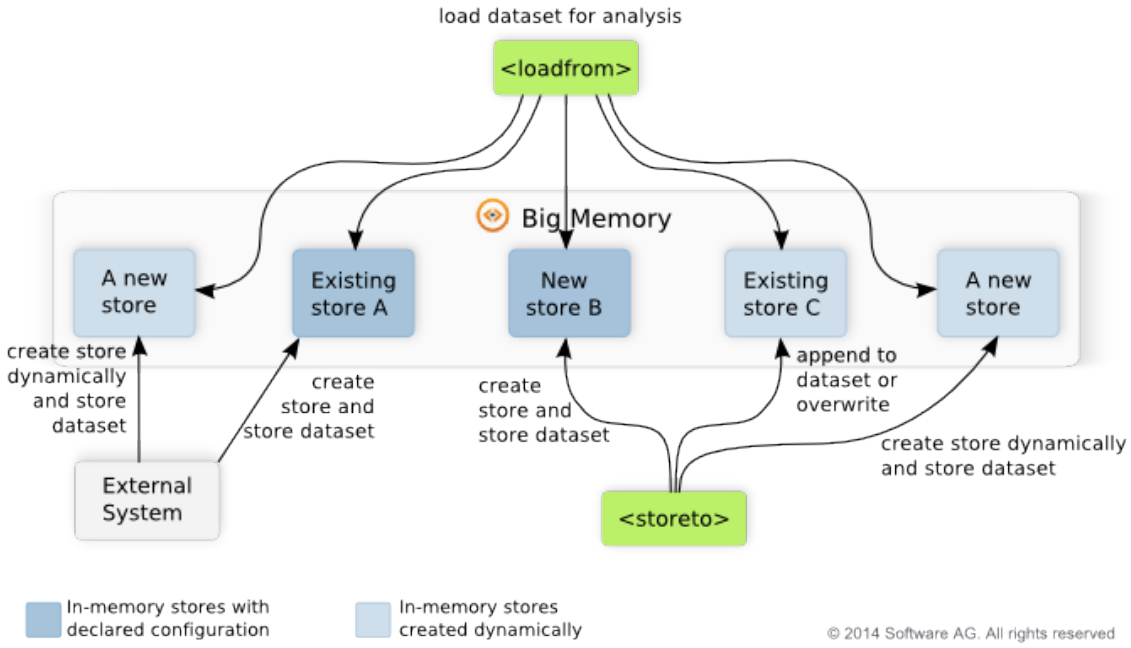
These In-Memory Stores allow you to store large datasets for quick access or retrieve them for analysis using RAQL. Getting Started with MashZone NextGen Analytics includes some very simple examples of how to do this. For additional information:

- See "Declared Versus Dynamic In-Memory Stores" on page 857 for an overview of the types of stores that are supported. See "External versus Internal Datasets" on page 859 for an overview of supported *producers*, the systems that store datasets.
- See "About BigMemory and the MashZone NextGen Analytics In-Memory Stores" on page 861 for information on how memory is managed, configured and deployed.
- See "Store Data in MashZone NextGen Analytics In-Memory Stores" on page 865 for information and examples of additional options to store datasets in stores.
- See "Load Data from the MashZone NextGen Analytics In-Memory Stores" on page 872 for information and examples of additional options to retrieve datasets from in-memory stores.

Declared Versus Dynamic In-Memory Stores

Each In-Memory Store holds one dataset with any number of rows, also known as entries, within the memory constraints defined for the store. In previous releases you could store many different datasets in one store with a dataset being one entry in the store.

As shown in the following figure, there are two ways to create In-Memory Stores: *declared* and *dynamic*.



Dynamic In-Memory Stores

Dynamic stores are created programmatically on the fly when:

- MashZone NextGen Analytics first stores a dataset to a store that does not yet exist.

  By default, MashZone NextGen Analytics sets the `overflowToOffHeap` property to true for dynamic stores. All other properties use their default values.

  You can set additional configuration properties to use with new dynamic stores created by MashZone NextGen Analytics in the dynamiccache.xml configuration file in the *MashZoneNG-config* /classes folder. For more information on these properties, see BigMemory documentation.

  In addition, each column in the datasets for dynamic In-Memory Stores created by MashZone NextGen Analytics is defined as searchable with the known or derived datatype for that column. For XML or CSV datasets, columns may all be considered string unless a schema is provided for the dataset. This also uses the MashZone NextGen Analytics class `de.rtm.adapters.bigmemory.RecordMetaDataAttributeExtractor`.

- An external system creates a store dynamically in BigMemory. These stores are also known as *dynamic external stores* because the external system is also the system the stores data in the store.

  Note: Access to dynamic external stores requires BigMemory be installed as a server or server array. MashZone NextGen must also have access to the BigMemory license.

To connect to dynamic external stores, a MashZone NextGen administrator must define connection configuration. The Terracotta Management Console (TMC) must also be running to successfully connect to a dynamic external in-memory store.

MashZone NextGen Analytics uses the connection configuration to retrieve configuration for the store from the Terracotta Management Console (TMC) that manages the BigMemory host for this store. This includes *search attribute* information that is *required* to allow MashZone NextGen Analytics to work with the columns in this dataset.

One common example is the use of a dynamic external store to allow MashZone NextGen to work with datasets from distributed stores in the Apama MemoryStore.

Declared In-Memory Stores

Declared stores are defined in BigMemory configuration files (ehcache.xml files) that MashZone NextGen administrators add to the MashZone NextGen Server to allow you to work with these stores. S

Declared In-Memory Stores can contain data loaded by MashZone NextGen or data loaded by external systems. With declared in-memory stores, the *producer* system that stores data in the store is the system that creates the store. MashZone NextGen Analytics is always the *consumer* system that retrieves data from the store for analysis.

Declaring stores before use allows you to:

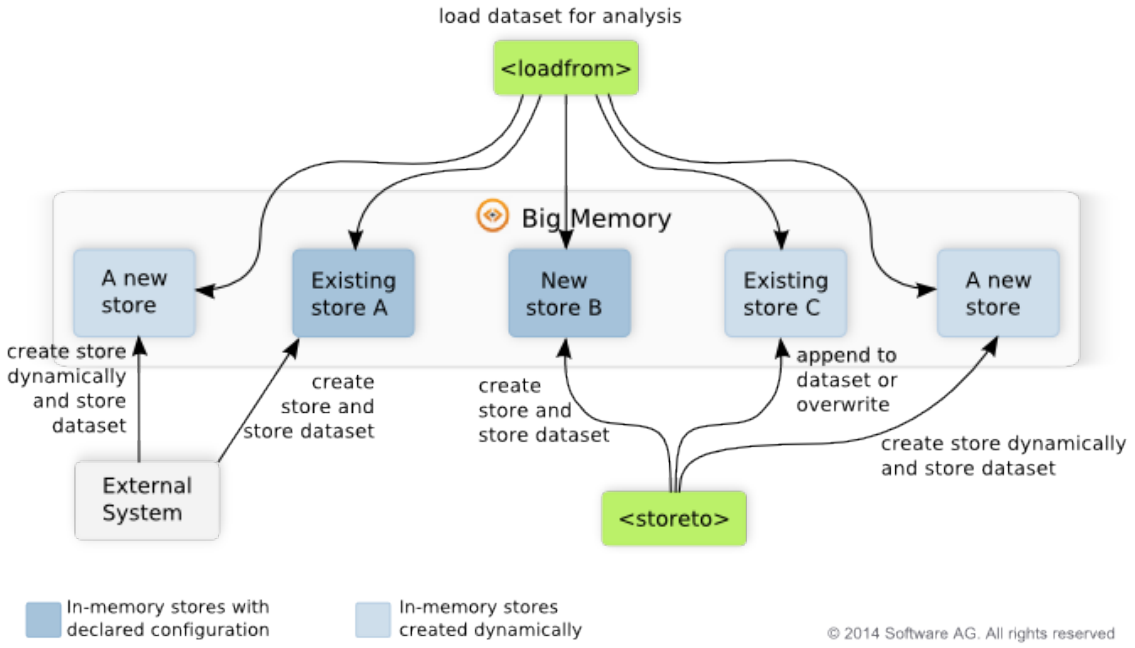
- Customize store properties for each store, providing better memory management and data retention.
- Define search attributes for the columns in the dataset in each store.

  Column search attributes are *required* to allow MashZone NextGen Analytics to work with datasets stored by external systems. It is optional, but a best practice for datasets stored by MashZone NextGen Analytics.

  Defining search information also allows MashZone NextGen Analytics to delegate filtering and sorting to the In-Memory Store which provides better performance.

External versus Internal Datasets

This figure also illustrates how datasets are stored in in-memory stores:



When MashZone NextGen Analytics stores data in an In-Memory Store, it appends the data to any existing data by default. You can also clear existing data before storing new data in the `<storeto>` statement.

Other systems can also store data in In-Memory Stores. This is *external data* and the store is sometimes called an *external in-memory store*.

In both cases, the system that first stores the dataset is the system that creates the In-Memory Store.

To allow MashZone NextGen Analytics to work with external data, external memory stores must be configured with:

- The name of the *cache manager* that manages memory for the in-memory store.

  Note: BigMemory does not require cache manager names, but they are a best practice for caches used as MashZone NextGen Analytics In-Memory Stores. Cache manager names prevent potential name collisions for stores.

- *Search attributes* for the dataset that identify the columns in the dataset and the datatype of the data in each column.

Configuration for declared stores specifies the cache manager, search attributes and connection information for the store.

With dynamic stores that hold external data, the external system must set a name for the cache manager and define search attributes programmatically when the external system creates the store, using the BigMemory API. See BigMemory Documentation for more information and examples.

Configuration defined in MashZone NextGen for the dynamic store allows MashZone NextGen Analytics to retrieve this configuration information.

About BigMemory and the MashZone NextGen Analytics In-Memory Stores

For information on how memory is managed for MashZone NextGen Analytics In-Memory Stores, see:

- "Initial Memory Configuration and Performance" on page 861
- "In-Memory Dataset Management" on page 861
- "Memory Management and Configuration" on page 861
  - "BigMemory Configuration, Cache Sizing and Pinning" on page 863

Initial Memory Configuration and Performance

Installation for MashZone NextGen uses a base configuration for a small web application for both the MashZone NextGen Server and the Integrated MashZone Server. This uses local memory only from the heap. In many cases, this base configuration is sufficient to work with MashZone NextGen in development environments with individual user computers.

You may need to adjust this configuration to provide adequate performance when you work with large datasets in MashZone NextGen Analytics, to use specific features in BigMemory or when you install MashZone NextGen in staging or production environments.

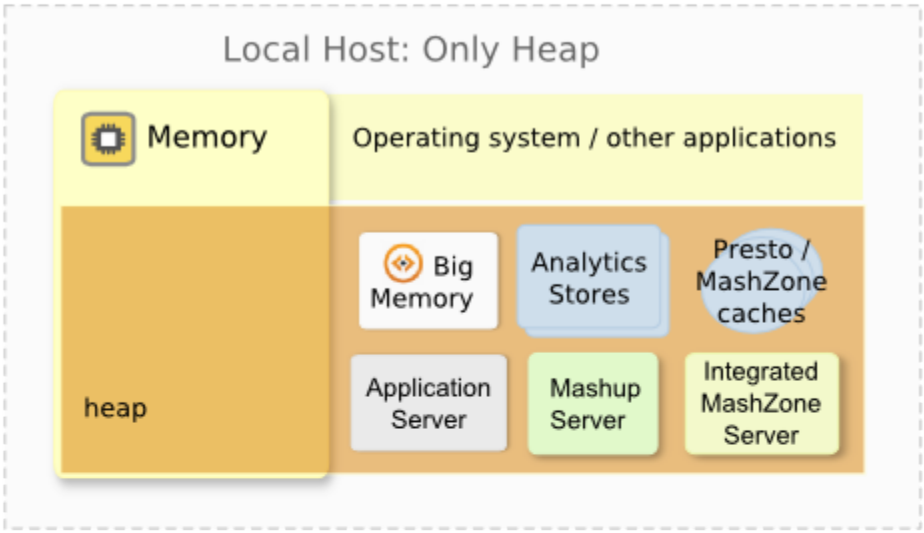
In-Memory Dataset Management

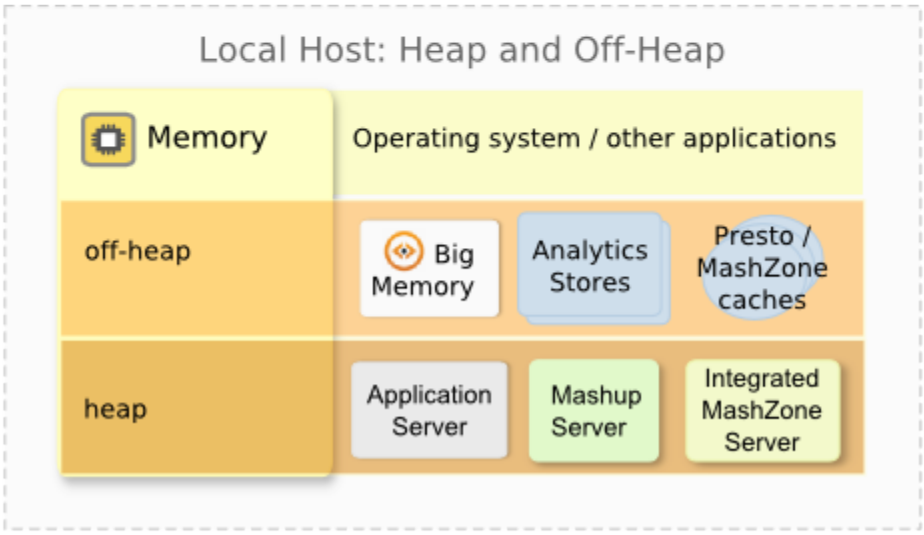
By default, the MashZone NextGen Analytics In-Memory Stores store datasets in memory with no expiration, so they remain in memory indefinitely. Stores use no persistence, however, so MashZone NextGen Server restarts clear all datasets.

In-Memory Stores also do not overflow to disk if the datasets you store exceed available memory. If the memory allocated to BigMemory is full, rows in datasets are evicted based on the least recent use to make room for new rows.

Memory Management and Configuration

BigMemory manages memory and data for both the MashZone NextGen Analytics In-Memory Stores and all caches for MashZone NextGen and the Integrated MashZone Server. The initial configuration for memory when you install MashZone NextGen uses only *heap* memory on the local host, as shown in this figure:





© 2014 Software AG. All rights reserved

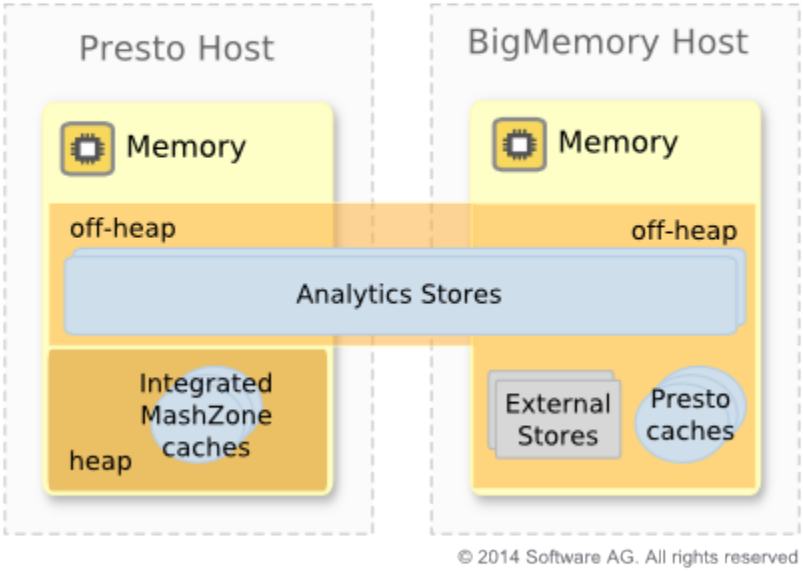
- *Heap* memory is configured for the JVM and used by MashZone NextGen, the Integrated MashZone Server, the Event Service and the application server that hosts MashZone NextGen. By default, heap is also used for In-Memory Stores and MashZone NextGen and MashZone caches.
- *Off-heap* memory is direct memory that is not controlled by the JVM. You can adjust MashZone NextGen memory configuration to make use of off-heap memory for both In-Memory Stores and MashZone NextGen caches, as shown in the previous figure.

  Note: Off-heap memory is only accessible if you have installed BigMemory Servers and added the BigMemory license to MashZone NextGen.

  Using off-heap memory can improve performance if available memory is *more than* the recommended minimum for MashZone NextGen Analytics.

- *Remaining direct memory* is used by the operating system and other applications running on the local host, such as browsers or other applications.

With BigMemory Servers and optional mirrors, memory is distributed as shown in the following figure:



MashZone NextGen, the Integrated MashZone Server and the Event Service use heap memory from the local host as usual, as do MashZone caches. Some local host memory is allocated for off-heap which is combined with a much large allocation of off-heap memory from the host for BigMemory.

The data for the MashZone NextGen Analytics In-Memory Stores and MashZone NextGen caches are distributed across both local and external off-heap memory both of which are managed by BigMemory. Off-heap memory managed by BigMemory can also contain external In-Memory Stores created by other systems. And of course some memory is allocated for the operating system or other applications on both the MashZone NextGen and BigMemory hosts.

You can also install BigMemory in a cluster to provide more memory capacity or support failover and other high availability features. .

Note: This is a high-level summary of memory use and storage tiers available with BigMemory. For more detailed information, see the BigMemory documentation.

How you configure memory depends on how MashZone NextGen and BigMemory are deployed:

BigMemory Configuration, Cache Sizing and Pinning

- **MashZone NextGen Default JVM Settings**

The MashZone NextGen default JVM settings are `-XX:PermSize=128M -XX:MaxPermSize=256M -Xms512M -Xmx1G`, 1 GByte Heap, no direct memory configuration supplied. The committed memory is 661MB.

The JVM settings are found in apache-tomee-jaxrs/bin/setenv.bat script.

> **Note:** There are no Garbage Collector settings specified. However the default Garbage Collector settings should work fine with Heap Size up to several GByte.

- **Default BigMemory Settings**

  The default settings for dynamic caches are `<ehcache name="dynamiccache" overflowToOffHeap="true" maxBytesLocalHeap="500M">`.

  Additionally each cache is configured with pinning=”incache”.

  Meaning:

  - `overflowToOffHeap` on CacheManager level has no effect.
  - BigMemory is actually never used with this default `configCaches` act as normal Java Objects. When the heap is full, you get an OoME.
  - Eviction policy is LRU (default). However no eviction should take place with this config, instead when there is no more space available, you'll get an OoME.

  > **Note:** If a CacheManager has a pool configured for using off-heap, the `overflowToOffHeap` attribute is automatically set to `true`.

  To use off-heap capabilities you have to install a Terracotta license. See "Manage Licenses for MashZone NextGen, Universal Messaging and BigMemory" on page 1572.

- **Notes on eviction**

  BigMemory is a cache and not a store. This means eviction is a central part of BigMemory . You can configure how eviction is done but you can never have a complete control over eviction. However with the appropriate configuration, BigMemory can be used as in-memory store.

  - Eviction takes place when there is not enough space in the cache and new data is coming. A cache has a certain size and thus number of max elements that can live in the cache. Older, less accessed elements get evicted if the number of elements in the cache exceeds the maximum.
  - Default eviction strategy is Least Recently Used.
  - Setting `ethernal=”true”` in a cache configuration does NOT prevent eviction.
  - Pinning has two options:

    1. `inCache` – Data is stored in any storage tier of the cache, depending on performance. Unexpired entries can never be evicted. If configured size is

exceeded, you get an OoME. Without an offheap configuration, everything lands in the heap and the limit it when the JVM is out of memory.

2. `localMemory` – Entries are evicted only when the store's configured size is exceeded. In a TSA scenario, entries are faulted from the server.

- Pinning is always local (L1 level), it is never distributed in a cluster

- **Means to overcome Eviction:**
  - **Size caches so data fits into them!** Leave 30% Room to avoid Throttling.
  - Locally you can overflow to disk(slow) or use TSA for better performance.
  - Pinning could be an option to override cache sizes but this is not best practice and you get OoME instead of eviction. At least makes the 'damage' visible.
- **See links for details:**
  - http://terracotta.org/documentation/4.1/bigmemorymax/configuration/data-life
  - "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664

Store Data in MashZone NextGen Analytics In-Memory Stores

The basics of storing a dataset in a new In-Memory Store created dynamically for this dataset is shown in "Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics" on page 696 in Getting Started. You can also:

- Store Data in Declared In-Memory Stores
- Set Unique Keys for Dataset Rows
- Append Query Results Repeatedly
- Clear Existing Data from In-Memory Stores
- Manage Memory for Dynamic In-Memory Stores

Store Data in Declared In-Memory Stores

Declared In-Memory Stores must be added as connections by MashZone NextGen administrators before you can load datasets from them.

Declared stores:

- May already exist and may already contain data that has been stored by other systems.

  MashZone NextGen can also store data in declared In-Memory Stores using `<storeto>`.

- Have configuration defined in an ehcache.xml file that defines the connection and behavior of the store. For example, declared stores may define search criteria or use different eviction policies.

- Have names in the form *data-source.store-name* . The data source name is assigned by the MashZone NextGen administrator to the configuration file and the store name is defined in the configuration for each store that is declared in the file.

The following example retrieves legislator data from a URL and stores this dataset in the sample In-Memory Store named `sample-cache.LegislatorsDeclCache`:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
xmlns:macro = "http://www.openmashup.org/schemas/v1.0/EMMLMacro"
name='LoadLegislatorsDeclarativeCache'>
  <output name='result' type='string'>
   Data loaded in cache. Use query-decl-cache.emml to query data from this cache.
  </output>
  <variable name="congressTupleStream" type="document" stream="true" />
  <variable name="congress" type="document"/>
  <directinvoke stream="true" outputvariable="congress"
    endpoint="http://mdc.jackbe.com/prestodocs/data/legislators.xml" />
  <raql query="select * from congress " stream="true"
    outputvariable="congressTupleStream" />
<storeto clearcache="true" cache="sample-cache.LegislatorsDeclCache"      key="firstname,lastname
</mashup>
```

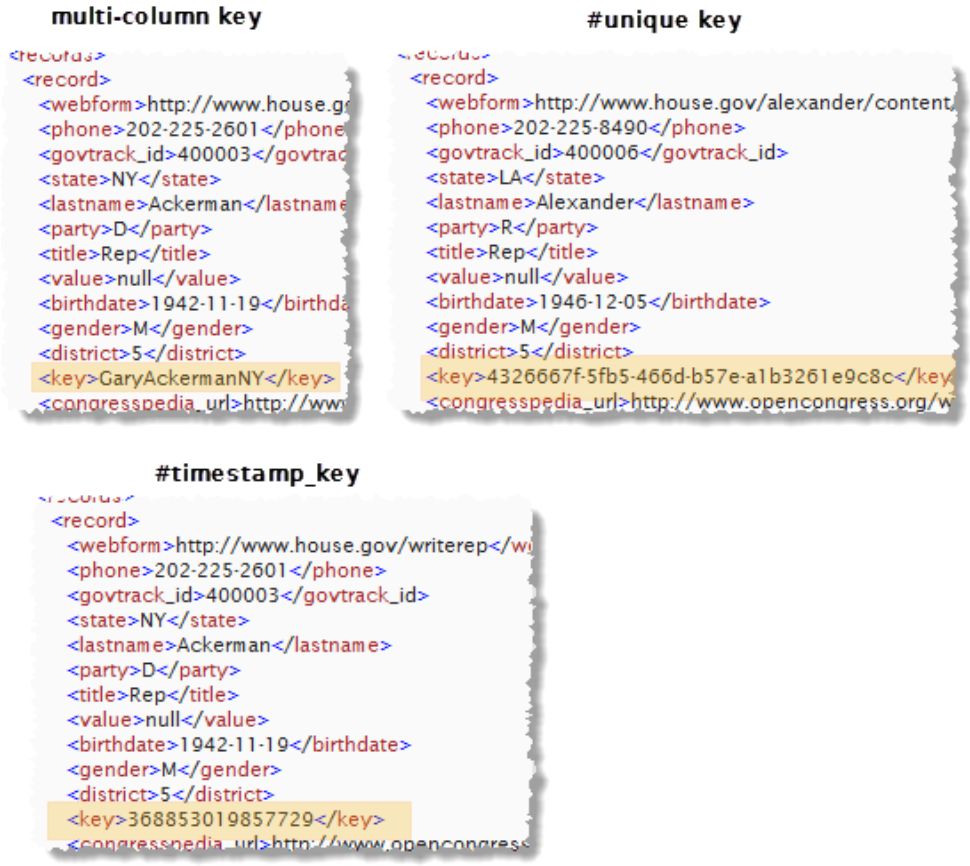
Some points to keep in mind in this example:

- The `clearcache` attribute is set to true to flush any existing data in this declared In-Memory Store.
- The `cache` attribute contains both parts of the In-Memory Store name:
  - Data source name,`sample-cache`, that was assigned to the configuration file that defines the connection to this In-Memory Store
  - The store name, `LegislatorsDeclCache`, that is defined in the configuration for this store.
- The `key` that is assigned to each row in this dataset combines the first name, last name and state of each legislator. This is an example of a multi-column key. See "Set Unique Keys for Dataset Rows" on page 866 for more information on different methods to assign unique keys.

Set Unique Keys for Dataset Rows

When you store datasets in an In-Memory Store, a `key` column is added and each row is assigned a unique key. You determine how this key is derived in the `key` attribute of `<storeto>`.

You can assign keys based on the time rows are added or a unique ID algorithm. You can also use an existing column in the dataset as the key or a combination of columns. For example:



See "<storeto>" on page 882 for detailed information on the methods to derive keys.

The example in Store Data in Declared In-Memory Stores combines multiple existing columns as the key. The example in Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics uses `#unique` to derive numeric keys.

Append Query Results Repeatedly

In cases where a source dataset is regularly updated, updates can be appended to the existing data in the In-Memory Store.

The following example mimics time-based updates by appending rows to an existing In-Memory Store using the EMML`<foreach>` statement. This statement loops through the Manufacturing Plants dataset (introduced in) and selects rows for a given country. These rows are then appended to an existing In-Memory Store named `storeAppendPlants`.

```
<!-- Loop for each country, retrieve plants for country and append to
     in-memory store, pause 5 seconds so timestamps are distinct -->
<foreach variable='location' items='$countries/countries/country' >
  <variable name="found" type="document" stream="true"/>
  <assign fromexpr="$location/text()" outputvariable="$searchFor"/>
  <directinvoke  method='GET' stream='true' outputvariable='plants'
    endpoint='http://mdc.jackbe.com/downloads/presto/data/mfgplants.csv' />

  <raql stream="true" outputvariable='found'>
    select Country, Name, Active_Production_Lines,
    Production_Lines_Under_Construction from plants
    where Country = '{$location}'
  </raql>
  <storeto cache='storeAppendPlants' key='#unique' variable='found' />
  <script type='text/javascript' outputvariable="result">
    <![CDATA[
      function pause(millis){
        var date = new Date();
        var curDate = null;

        do { curDate = new Date(); }
        while(curDate-date < millis);
      }
      pause(5000);
    ]]>
  </script>
</foreach>
```

The example then pauses to ensure that the rows appended are distinct based on their timestamp, as shown in this example of the results:

Some points to keep in mind in this example:

- The first loop of this mashup creates the In-Memory Store the first time the mashup is run.
- Every row stored has a column added with the timestamp when the row was added. For more information on timestamps in the In-Memory Store, see "About Row Timestamps for Stored Datasets" on page 870.
- The JavaScript function in this example is used solely for effect, to ensure distinctly different timestamps for each append to the In-Memory Store.

> **Note:** This In-Memory Store is also used in the Example 130. Load Dataset Rows for Specific Time Periods example to illustrate loading data based on row timestamps.

The complete EMML code for this example is shown here.

> **Note:** Be aware that this mashup will take over 50 seconds to run because of the deliberate pauses.

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  name='storeAppend'>
  <output name='result' type='document' />
  <variable name="plants" type="document" subType="csv"/>
```

```
  <!-- Countries to interate over -->
  <variable name="countries" type="document">
    <countries>
      <country>BELGIUM</country>
      <country>CANADA</country>
      <country>FRANCE</country>
      <country>INDIA</country>
      <country>ITALY</country>
      <country>JAPAN</country>
      <country>NETHERLANDS</country>
      <country>SPAIN</country>
      <country>SWEDEN</country>
      <country>UNITED KINGDOM</country>
    </countries>
  </variable>
  <variable name="searchFor" type="string" />
  <!-- Loop for each country, retrieve plants for country and append to
       in-memory store, pause 5 seconds so timestamps are distinct -->
  <foreach variable='location' items='$countries/countries/country' >
    <variable name="found" type="document" stream="true"/>
    <assign fromexpr="$location/text()" outputvariable="$searchFor"/>
    <directinvoke  method='GET' stream='true' outputvariable='plants'
      endpoint='http://mdc.jackbe.com/prestodocs/data/mfgplants.csv' />
    <raql stream="true" outputvariable='found'>
      select Country, Name, Active_Production_Lines,
      Production_Lines_Under_Construction from plants
      where Country = '{$location}'
    </raql>
    <storeto cache='storeAppendPlants' key='#unique' variable='found' />
    <script type='text/javascript' outputvariable="result">
      <![CDATA[
        function pause(millis){
          var date = new Date();
          var curDate = null;
          do { curDate = new Date(); }
          while(curDate-date < millis);
        }
        pause(5000);
      ]]>
    </script>
  </foreach>
</mashup>
```

About Row Timestamps for Stored Datasets

In addition to keys, RAQL adds a timestamp to each row of a dataset for the date and time that row was added to the In-Memory Store. In some cases, datasets already have a column with a timestamp related to the original source of the data.

Because of potential name conflicts, the actual column name for the In-Memory Store timestamp, *varies* based on the dataset in question and the storage mode (append or overwrite):

If Existing Dataset	And Store Mode is	In-Memory Store Timestamp Column
Has *no* column named timestamp.	Overwrite or append.	`timestamp`

If Existing Dataset	And Store Mode is	In-Memory Store Timestamp Column
Has a column named `timestamp`.	Append rows to an existing dataset key.	`timestamp` In this case, the timestamp for the In-Memory Store will overwrite the original source timestamp, unless you provide an alias for the original column.
	Overwrite existing dataset.	`_timestamp`

Clear Existing Data from In-Memory Stores

By default, storing a dataset simply appends the data to any existing data in an In-Memory Store. You can clear all existing data and add the current dataset by setting the `clearcache` attribute to true. See "Store Data in Declared In-Memory Stores" on page 865 for an example.

Manage Memory for Dynamic In-Memory Stores

When you store a dataset in an In-Memory Store for the first time with a <storeto> statement, MashZone NextGen Analytics creates the In-Memory Store in either local memory or in memory managed by a BigMemory server or server array.

If the store is a declared store, the configuration added to MashZone NextGen when the store was declared defines where the store may be kept in memory.

For dynamic stores, the In-Memory Store store is created in local memory by default. You can configure the store to allow a BigMemory server to keep the store in memory in the BigMemory server or server array using:

- `useTSA = "true"` on the `<storeto>` statement in the mashup that creates this In-Memory Store. For example:

```
...
<storeto useTSA="true" cache='myEvents' key='#unique' variable='theseEvents' />
...
```

- Configuration in the dynamiccache.xml file in the *webapps-home* /presto/WEB-INF/classes folder. You can use this configuration file to define properties for all In-Memory Stores created dynamically by MashZone NextGen.

  Include the connection information to the BigMemory Server that manages memory for In-Memory Stores. For example:

```
<ehcache name="dynamiccache" overflowToOffHeap="true"
```

```
   maxBytesLocalHeap="1G">
  <terracottaConfig url="localhost:9510" /></ehcache>
```

Load Data from the MashZone NextGen Analytics In-Memory Stores

To load data from MashZone NextGen Analytics In-Memory Stores, datasets must first be stored using `<storeto>` in MashZone NextGen mashups or stored by external systems. See "Store Data in MashZone NextGen Analytics In-Memory Stores" on page 865 for more information and examples.

The basics of using `<loadfrom>` to load a dataset stored in an In-Memory Store that MashZone NextGen Analytics created dynamically are covered in "Group and Analyze Rows" on page 699 and "Group and Analyze Rows with Row Detail" on page 701 in Getting Started.

You can also:

- Example 128. Load a Dataset from a Declared In-Memory Store
- Example 129. Load Datasets from a Dynamic External In-Memory Store
- Example 130. Load Dataset Rows for Specific Time Periods
- Example 131. Use Pagination to Fine Tune Loading Performance
- Example 132. Handle Missing In-Memory Stores

Load a Dataset from a Declared In-Memory Store

Loading data from a declared In-Memory Store is identical to loading data from a dynamic store except for the form of the store name. Declared store names are compound including both the data source name and the store name.

The following example loads data stored in a declared store named `sample-cache.LegislatorsDeclCache`. The data source name is `sample-cache` and the store name is `LegislatorsDeclCache`. See "Store Data in Declared In-Memory Stores" on page 865 for an example of the mashup that stores data in this In-Memory Store.

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd'
    xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
    xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
    name='congressByStateGender'>
  <output name='result' type='document' />
  <variable name="congress" type="document"/>
  <loadfrom cache='sample-cache.LegislatorsDeclCache' variable='congress' />
  <raql outputvariable='result'>
    select firstname, lastname, state, party, gender from congress
    order by state, gender
  </raql>
</mashup>
```

Load Datasets from a Dynamic External In-Memory Store

With dynamic In-Memory Stores that are created and data is stored by external systems, there is no configuration file to provide connection information or information on the data in the dataset. Instead, MashZone NextGen administrators must add configuration

for external dynamic stores to allow MashZone NextGen Analytics to connect to the store and allow RAQL to work with the dataset.

Once a connection has been made, you can load data from dynamic external stores just like any store, except for the form of the store name. Store names for dynamic external stores are compound names including both the connection name and the store name (the cache name).

Important: In addition to the in-memory store, the Terracotta Management Console (TMC) must also be running to successfully connect to a dynamic external in-memory store.

The following example loads data stored in a dynamic external store named `ApamaExternalBigMemory.SampleDynamicCache`.

The connection name is `ApamaExternalBigMemory`. The store name assigned by the external system when it created the store is `SampleDynamicCache`.

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
      xmlns:macro = "http://www.openmashup.org/schemas/v1.0/EMMLMacro"
      name='SampleOfExternalCache'>
  <output name='result' type='document'/>
  <variable name="simpleData" type="document" stream="true"/>
  <loadfrom cache="ApamaExternalBigMemory.SampleDynamicCache"
    variable="$simpleData" />
  <raql outputvariable="result" >
      select * from simpleData
  </raql>
</mashup>
```

Load Dataset Rows for Specific Time Periods

In many cases, analysis is only interested in the most recent updates to a dataset. You can load specific rows from a dataset stream stored in an In-Memory Store based on a time interval starting backwards from now, using the `period` attribute in the `<loadfrom>` statement.

You specify the recent time period for the rows you want to load as a number of seconds, minutes, hours, days or weeks. See "<loadfrom>" on page 877 for the syntax to use.

The following example works in conjunction with the example shown in "Append Query Results Repeatedly" on page 867 to illustrate the effect of retrieving dataset rows based on a recent time period:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas
/EMMLPrestoSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  xmlns:presto='http://www.jackbe.com/v1.0/EMMLPrestoExtensions'
  name='loadLastPeriod'>

  <output name='result' type='document' />
  <variable name="lastPeriod" type="document" stream="true"/>
  <loadfrom cache='storeAppendPlants' variable='lastPeriod' period='10m'/>
  <raql outputvariable='result'>
    select distinct Country, timestamp from lastPeriod
  </raql>
</mashup>
```

It retrieves all rows from the `storeAppendPlants` in-memory store that were added within the last period of time set in the `period` attribute and then selects the list of distinct countries within those rows and the timestamp. The following examples of results for different time periods shows which rows were loaded:

Results for Last 5 Minutes

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
  <record>
    <Country>SWEDEN</Country>
    <timestamp>Fri Sep 20 15:51:29 PDT 2013</timestamp>
  </record>
  <record>
    <Country>UNITED KINGDOM</Country>
    <timestamp>Fri Sep 20 15:51:35 PDT 2013</timestamp>
  </record>
  <record>
    <Country>SPAIN</Country>
    <timestamp>Fri Sep 20 15:51:24 PDT 2013</timestamp>
  </record>
  <record>
    <Country>JAPAN</Country>
    <timestamp>Fri Sep 20 15:51:14 PDT 2013</timestamp>
  </record>
  <record>
    <Country>NETHERLANDS</Country>
    <timestamp>Fri Sep 20 15:51:19 PDT 2013</timestamp>
  </record>
</records>
```

Results for Last 10 Minutes

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
  <record>
    <Country>SWEDEN</Country>
    <timestamp>Fri Sep 20 15:51:29 PDT 2013</timestamp>
  </record>
  <record>
    <Country>BELGIUM</Country>
    <timestamp>Fri Sep 20 15:50:47 PDT 2013</timestamp>
  </record>
  <record>
    <Country>UNITED KINGDOM</Country>
    <timestamp>Fri Sep 20 15:51:35 PDT 2013</timestamp>
  </record>
  <record>
    <Country>SPAIN</Country>
    <timestamp>Fri Sep 20 15:51:24 PDT 2013</timestamp>
  </record>
  <record>
    <Country>FRANCE</Country>
    <timestamp>Fri Sep 20 15:50:58 PDT 2013</timestamp>
  </record>
  <record>
    <Country>JAPAN</Country>
    <timestamp>Fri Sep 20 15:51:14 PDT 2013</timestamp>
  </record>
  <record>
    <Country>INDIA</Country>
    <timestamp>Fri Sep 20 15:51:03 PDT 2013</timestamp>
  </record>
```

To try this example, use the following EMML code for this `loadLastPeriod` mashup and open it in Mashup Editor. Also create or open the `storeAppend` mashup shown in Append Query Results Repeatedly. Then run `storeAppend`. Wait a few seconds or

minutes, update the value of `period` in `loadLastPeriod` and run it to see which rows it loads.

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  name='loadLastPeriod'>
  <output name='result' type='document' />
  <variable name="lastPeriod" type="document" stream="true"/>
  <loadfrom cache='storeAppendPlants' variable='lastPeriod' period='5m' />
  <raql outputvariable='result'>
    select distinct Country, timestamp from lastPeriod
  </raql>
</mashup>
```

Use Pagination to Fine Tune Loading Performance

By default, datasets are streamed using a default *fetch size* to define how many rows are loaded at a time. Aggregate calculations are not completed until the entire dataset is loaded.

You can fine tune the page size used with the `fetchsize` attribute. This can improve performance during loadng. For example:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd'
    xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
    xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
    name='congressByStateGender'>
  <output name='result' type='document' />
  <variable name="congress" type="document"/>
  <loadfrom cache='sample-cache.LegislatorsDeclCache'
fetchsize='10000' variable='congress' />
  <raql outputvariable='result'>
    select firstname, lastname, state, party, gender from congress
    order by state, gender
  </raql>
</mashup>
```

Handle Missing In-Memory Stores

Since In-Memory Stores can be created on demand, errors can occur in mashups that load datasets if the store is missing unexpectedly. One way to avoid this is to have a mashup optionally store the dataset if it is not present using the <try> and <catch> statements in EMML.

The following example uses a memoization pattern to try to load the dataset and if not found then load the data from an original data source, store it and continue the analysis.

Note: This example uses data from an Atom feed from the USGS on recent earthquakes.

The EMML and RAQL code for this single mashup is shown here:

```
<mashup xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='http://www.openmashup.org/schemas/v1.0/EMML/../schemas/EMMLSpec.xsd'
  xmlns='http://www.openmashup.org/schemas/v1.0/EMML'
  xmlns:macro='http://www.openmashup.org/schemas/v1.0/EMMLMacro'
  name='tryLoadCatch'>
```

```
  <output name='result' type='document' />
  <variable name="quakesDS" stream="true" type="document"/>
  <try>
    <loadfrom cache="quakeLastWeek" variable="quakesDS"/>
    <raql outputvariable="result">
      select title, date, lat, long from quakesDS
    </raql>
    <display message="dataset found"/>
    <catch type='EMMLException e'>
      <variable name="quakeSrc" type="document"/>
      <variable name="quakeStore" stream="true" type="document"/>
      <directinvoke
        endpoint='http://earthquake.usgs.gov/earthquakes/feed/atom/4.5/week'
        method='GET' stream='true' outputvariable='quakeSrc'/>
      <raql outputvariable="quakeStore" stream="true">
        link.href as link,
        split_part(point,' ','1') as lat,
        split_part(point,' ','2') as long,
        concat(category.term,' magnitude') as mag from quakeSrc/feed/entry
      </raql>
      <storeto cache="quakeLastWeek" key="#unique" variable="quakeStore"/>
      <display message="key stored"/>
      <loadfrom cache="quakeLastWeek" variable="quakesDS"/>
      <raql outputvariable="result">
        select title, date, lat, long from quakesDS
      </raql>
    </catch>
  </try>
</mashup>
```

The <try> loop tries to retrieve the dataset stream from an existing In-Memory Store. If the store already exists, it executes a simple RAQL query.

The <catch> portion of the loop will catch the exception thrown if the store is not found in the <try> loop. It then invokes the REST mashable, queries the results and loads the dataset to the store.

If you run this mashup in the Mashup Editor, look at the Console section to see the messages output from the <display statements in the <try> and <catch> loops. The first time you run the mashup you should see the <catch> message. Run it again and you should see the message from <try>.

RAQL Extension to EMML Statements for Mashups

The Real-Time Analytics Query Language adds the following extension statements to EMML for use in mashups:

- <loadfrom>
- <raql>
- <snapshot>
- <storeto>

RAQL also adds some specific attributes to existing EMML statements to:

- Provide streaming access to data. See "RAQL Extensions to EMML Statements for Streaming" on page 885.
- Support files as a data source for variables in EMML. See "RAQL Extensions to <variable> for File Data Sources" on page 886 for information.
- Supports metadata schemas for XML or CSV datasets to simplify row definitions, provide column metadata or other options for RAQL queries. See "RAQL Extensions for Dataset Schemas" on page 887 for information.

<loadfrom>

Loads a dataset from one named In-Memory Store into the specified variable. Data is streamed and may only load a specific set of rows from the dataset based on a time period specified.

Important: The semantics of this statement have changed significantly in this release fromMashZone NextGen version 3.6. The `version` attribute identifies which semantics are used.

For information on 3.6 semantics and the available attributes, see<loadfrom> forMashZone NextGen3.6.

For examples of this statement see "Load Data from the MashZone NextGen Analytics In-Memory Stores" on page 872.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
version		The version of RAQL that is used in this statement. This version affects what attributes are required and the meaning of some attributes in <loadfrom>. Possible values are: - `1.0` = the initial version of RAQL released in MashZone NextGen 3.6. See <loadfrom> forMashZone NextGen3.6 for information on the available attributes when this version is in effect.

Name	Required	Description
		■ `2.0` = the default. This is the current version of RAQL.
cache	yes	The name of the In-Memory Store where the dataset to load resides. **Note:** Store names can be a literal value or you can assign them dynamically using mashup expressions. See "Dynamic Mashup Expressions" on page 604 for more information. For dynamic In-Memory Stores, this name is a simple string. For declared In-Memory Stores, this name is in the form *data-source-name.store-name* . See "Declared Versus Dynamic In-Memory Stores" on page 857 for more information.
variable	yes	The name of the variable to hold the dataset or specific rows to retrieve. This variable *must* have been explicitly declared previously in the mashup.
period		Defines an optional time period used to retrieve specific rows in the dataset based on their timestamp. Valid time periods include: ■ *n* `s` = the last *n* seconds ■ *n* `m` = the last *n* minutes ■ *n* `h` = the last *n* hours ■ *n* `d` = the last *n* days ■ *n* `w` = the last *n* weeks ■ *n* = with no period specified, this defaults to the last *n* minutes See "Example 130. Load Dataset Rows for Specific Time Periods" on page 873 for an example.
fetchsize		The numbers of rows to request from the In-Memory Store at one time during loading. This can be used to fine tune performance with very large datasets or datasets with a large number of columns per row.

<raql>

Executes a RAQL query against the variable, and optional path, defined in the From clause of the query.

See "The RAQL Query Syntax" on page 692 and "Getting Started with MashZone NextGen Analytics" on page 689 for general information and examples. See "RAQL Queries" on page 704 for links to examples of the syntax variations for each clause that you may use.

Can Contain	The RAQL query to execute, unless the query is specified in the `query` attribute.
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
query	conditional	The RAQL query to execute. This query may use the dataset stream from the `snapshotquery` or from the variable, and optional path, specified in the From clause. You *must* specify the RAQL query either as a simple text inside the `<raql>` element or in this attribute. See "The RAQL Query Syntax" on page 692 for general information. See "RAQL Queries" on page 704 for links to examples of the syntax variations for each clause that you may use.
snapshotquery		An optional SQL query to execute to retrieve snapshots from the MashZone NextGen Snapshot Repository. The results of this query then become the dataset stream used in the RAQL query. Specifying a snapshot query in this attribute creates a dataset stream in an implicit variable named `snapshots` that is local to this `<raql>` statement. You can instead created a named variable for snapshot datasets using the "<snapshot>" on page 880 extension statement.

Name	Required	Description
		See "<snapshot>" on page 880 for information on valid SQL queries for MashZone NextGen snapshots. See "Load Snapshot Data with <snapshot> or <raql>" on page 714 for examples.
condition		Defines a condition that must be met to run this query. This is equivalent to using <raql> within an EMML <if> statement.
stream		■ If `true`, the query output remains a dataset stream that can be used in other RAQL statements, such as another `<raql>` statement or `<storeto>`. ■ If `false` (the default) or if omitted, the query output becomes a document type variable that can be used in other EMML statements including `<output>` to define the results of the mashup. You can also define streaming access on the `<variable>` used as the output variable. See "The Stream/Document Boundary" on page 695 for more information.
outputvariable		The name of the variable to receive the results of this query. This must be a document-type variable. If this variable has already been declared with `stream="true"`, query results use streaming to populate this variable. See "RAQL Extensions to EMML Statements for Streaming" on page 885 for more information.

<snapshot>

Loads a dataset stream with the results of a query to the MashZone NextGen Snapshots Repository. This dataset stream contains a collection of snapshots of results for:

- The specified mashable or mashup.
- The specified mashable and operation.
- A specified time period.
- A scheduled snapshot job.

This statement also implicitly creates the variable to hold the dataset stream.

You can also retrieve snapshots anonymously, without creating a named variable to hold the dataset stream, using <raql>. See "Load Snapshot Data with <snapshot> or <raql>" on page 714 for examples.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
name	yes	The name to assign to the variable to hold the snapshot collection from this query. This variable automatically is configured for streaming.
query	yes	The query to execute to retrieve a collection of snapshots for a specific mashable or mashup, a specific mashable operation, a specific date and time or a specific snapshot job. This SQL query *must* include a where clause to identify the specific snapshots to include in this dataset. You can use the following columns in a snapshot query: ■ `service` = the mashable or mashup ID for the snapshots to return. **Note:** The ID for a mashable or mashup is different from the name. To find a mashable or mashup ID, open the mashable or mashup and click **Show** > **Info**. ■ `operation` = the name of the operation for the snapshots to return. This is required if you use `service` and the mashable has multiple operations. ■ `createdtime` = a timestamp when the snapshot was taken.

Name	Required	Description
		■ `job` = the job ID for the snapshots taken by a scheduled snapshot job.

<storeto>

This statement stores the dataset contained in the specified variable in the specified In-Memory Store. This can be a declared In-Memory Store with a connection that a MashZone NextGen administrator has already added or it can be a dynamic In-Memory Store created by MashZone NextGen.

Important: The semantics of this statement have changed significantly from MashZone NextGen version 3.6. Effective with version 3.8, the original semantics are *no longer* supported.

If data already exists in this store, the dataset in this statement is appended to existing data by default. You can choose to clear any existing data, if needed. You also choose the method used to assign unique keys to rows in this dataset and control some aspects of memory management for this store.

For examples, see "Store Data in MashZone NextGen Analytics In-Memory Stores" on page 865.

Can Contain	Empty
Allowed In	mashup | catch | else | elseif | for | foreach | if | macro | operation | sequence | try | while

Attributes

Name	Required	Description
version		The version of RAQL that is used in this statement. Version `2.0` is the default and is the only semantics that are supported. **Important:** The semantics for version 1.0, from MashZone NextGen 3.6 are *no longer* supported.
key	yes	The method that MashZone NextGen Analytics uses to assign unique keys to each row in this dataset. Valid methods include:

Name	Required	Description
		■ `#auto_sequence` = reserved for future use. ■ *column1[,column2,...]* = one or more column names within this dataset. Multiple column names should be separated with commas. If a single column is provided, the values in this column must be unique in the dataset. If multiple columns are provided, the combination of values for the columns must be unique. These values are concatenated to define the key value. If a column name is unknown, an error message will be displayed. ■ `#timestamp_key` = a unique number based on the current timestamp down to nano seconds (`Systsem.nanoTime` in Java). ■ `#unique` = generates a universal unique ID (`java.util.UUID` in Java) for each row. You may also use `#uuid`. ■ default is '#unique'. If the attribute is omitted, the value will set to default.
cache	yes	The name of the In-Memory Store to add this data to. This can be a declared store, using a name in the form *data-source-name.store-name* , or a dynamic store using a simple name. For more information, see "Declared Versus Dynamic In-Memory Stores" on page 857. **Note:** Store names can be a literal value or you can assign them dynamically using mashup expressions. See "Dynamic Mashup Expressions" on page 604 for more information. If this In-Memory Store already exists, the dataset is appended to existing data by default. See the `clearcache` attribute for more information.
variable	yes	The name of the variable containing the data to be stored.
searchattributes		A comma-separated list of the columns in this dataset that should be searchable. If omitted,

Name	Required	Description
		MashZone NextGen makes all columns in this dataset searchable.
clearcache		Whether existing data in the In-Memory Store should be deleted. This defaults to `false`, appending data to any existing data.
useTSA		Whether this In-Memory Store should be stored solely in local MashZone NextGen memory (`false`) or can be stored in memory managed by an external BigMemory Server or server array (`true`). Local memory storage is the default. Memory management for In-Memory Stores is typically defined: ■ By configuration for declared In-Memory Stores or ■ Programmatically for In-Memory Stores that are created dynamically by external systems Thus this property is only relevant for In-Memory Stores that are dynamically created by MashZone NextGen Analytics. Configuration in the dynamiccache.xml file in *webapps-home* /presto/WEB-INF/classes folder defines the connection information for the BigMemory Server.

<dropcache>

Drop cache command removes a In-Memory store/cache. All cache contents along with any cache configuration(search attributes, column types) are erased.

```
<dropcache cache="<name>"/>
```

A distributed cache located in BigMemory Terracotta Server Array (TSA), must be removed using cache removal command in Terracotta Management Console (TMC). Please see the BigMemory TMC documentation for details.

Attributes

Name	Required	Description
cache	yes	The name of the In-Memory Store/cache that is to be removed.

RAQL Extensions to EMML Statements for Streaming

The `stream` attribute is available in the following EMML statements for use with MashZone NextGen Analytics:

- <directinvoke>
- <raql>
- <sql>
- <variable>

Note: Streaming access is always enabled for <loadfrom> and <snapshot> statements.

This attribute may be `true` or `false`. If omitted, it defaults to false.

Data Access as Documents with No Streaming

If the `stream` attribute is omitted in EMML statements or it is set to `false`, results from `<directinvoke>`, `<raql>` or `<sql>` statements are not available to other mashup statements until all results have been received. For example:

```
<directinvoke method="GET" outputvariable="$result" endpoint="http://
www.myCompany.com/rest-services/getNames"/>
```

The results are also represented in XML, creating a document type variable with a *document object model* (a DOM) which takes time as well as additional memory. With very large datasets, this can cause the mashup to fail due to inadequate memory or to timeout unnecessarily. The DOM, however, does enable access to the data in EMML statements using XPath rather than RAQL.

Data Access as Streaming Datasets

Datasets loaded with the `<loadfrom>` extension statement always use streaming, making some data available before all results have been received. Streaming also prevents the creation of a document object model (DOM). For examples, see "Group and Analyze Rows" on page 699 and "Group and Analyze Rows with Row Detail" on page 701 in Getting Started.

Datasets loaded with `<directinvoke>`, `<sql>` or <variable> statements also use streaming *if* `stream = "true"`. For examples, see "A Basic RAQL Query" on page 689 and in Getting Started and "Load Data with <sql>" on page 710.

Query results from a `<raql>` extension statement also can use streaming to populate the output variable if `stream = "true"`. For an example, see "Use an In-Memory Store to Store and Load Datasets for MashZone NextGen Analytics" on page 696 in Getting Started. Storing datasets to the In-Memory Store with `<storeto>` also always uses streaming.

There are two *critical* differences to keep in mind when accessing streamed data in a mashup:

- Streamed data requires RAQL to access the data. Since no DOM exists, the data is not accessible in other EMML statements using XPath.
- The scope for mashup variables that are used to hold streamed data is limited to *one*EMML extension statement. Data is streamed to the receiving statement and then *discarded*.

  There are a few ways to handle streamed access when a mashup needs to process a stream in several statements:

  - Save the streamed dataset as a document-type variable with a DOM and use the DOM with EMML statements.
  - Load the dataset as a stream multiple times in a mashup.
  - Separate each process of the dataset stream into different mashups and call these mashups in another mashup.

Stream Partitions

Data is streamed in *partitions* that define the maximum number of rows that are transmitted as a group. This also determines the maximum memory that MashZone NextGen needs to handle the stream as only one partition is in active memory at a time. RAQL uses a default partition of 10,000 rows.

RAQL Extensions to <variable> for File Data Sources

The `datafile` attribute is available in the `<variable>` statement in EMML for use *solely* in RAQL queries. This attribute allows a mashup to directly populate the variable with the contents of the specified file, *without* registering the file as a MashZone NextGen mashable.

Files used as data sources must:

- Contain data in a format supported by MashZone NextGen Analytics. Currently this includes CSV (comma-separated-values) or XML. See "Supported Data Formats for RAQL" on page 771 for details.

  Note: Because the document in a variable populated from a file is not necessarily XML and may not have a DOM, these variables can generally only be used in `<raql>` statements. Other EMML statements cannot work with this data.

- Be located in the classpath for the MashZone NextGen Server. You can place data source files in *web-apps-home* /presto/WEB-INF/classes or any folder outside of MashZone NextGen that you choose to add to the classpath.

You specify the file name in `datafile` as a literal or dynamic value. For an example of datasets loaded directly from a file using a literal name, see "A Basic RAQL Query" on page 689.

You can also supply file names dynamically from input parameters or other variables. For example:

```
<mashup name="FileAsInput"
  xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
  xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro"
  xmlns:presto="http://www.jackbe.com/v1.0/EMMLPrestoExtensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openmashup
    <output name="result" type="document"/>
    <input name="srcFile" type="string"/>
    <variable datafile="$srcFile" name="srcData" stream="true"
      type="document"/>
    <raql outputvariable="result">
      select * from srcData limit 25
    </raql>
</mashup>
```

RAQL Extensions for Dataset Schemas

Two types are available in the `<variable>` statement in EMML for use *solely* in RAQL queries:

- `schema` is the type to use variables that declare a schema for a dataset used in RAQL. Schemas may be defined directly in a mashup or they can be defines as global MashZone NextGen attributes for use in several mashups.
- `variable:`*schema-type-variable-name* is the type to use for variables that contain datasets used in RAQL when a schema for that dataset has been declared in the mashup or is defined in a MashZone NextGen global attribute. The *schema-type-variable-name* is the name of an existing variable with `type="schema"` that contains or references the schema for this dataset.

Variables that declare a schema provide metadata for a dataset including datatype information for columns, path information to the elements that should be treated as dataset rows and other options for datasets in the CSV format. For information about the schema syntax for datasets and examples of schemas, see "Providing Dataset Path and Datatype Information in a Schema" on page 722.

RAQL Samples

With theMashZone NextGeninstallation some additional samples are installed.

You can install the samples located in *MashZoneNG-install* /samples/emml/raql using the *MashZoneNG-install* /prestocli/bin/publish-mashups scripts. These scripts will install any of the samples located in *MashZoneNG-install* /samples/emml and its subfolders.

After installation, the samples are shown in the Mashup Editor under **My Mashups** and can be opened directly from there.

Sample name	Short Description
cache-load-store.emml	Cache initialization logic using <storeto/> and <loadfrom/> for subsequent RAQL queries.
correlate1.emml	RAQL query to correlate 2 column values (stock volume & closing price) grouped by months. Uses group aggregation to get only summary values; Uses windows aggregation to get summary values along-with record details.
correlate2.emml	Correlation of Stock Closing price & Volume over 3-month Periods.
covariance.emml	Computes Covariance of 2 columns(Stock closing price, volume) for each stock symbol.
cube-rollup.emml	Usage of ROLLUP, CUBE and GROUPING SETS to aggregate data on multiple levels
date-num-funcs.emml	Showcases Conversion and Locale-based Formatting functions : Currency formatting, Number formatting, Percent formatting, Integer formatting with group separators, Data to/from Epoc conversions. Date Formatting.
datefmts.emml	Use of to_date(...) date conversion function
di-cache-stream.emml	Invokes REST service URL endpoint and stores data into a BigMemory declarative cache.
dom-cache.emml	Stores XML DOM in cache using <storeto/>; then retrieves this DOM

Sample name	Short Description
	from Cache using <loadfrom/> for querying using RAQL .
except.emml	Usage of RAQL except operator to find difference between 2 datasets
hierarchy-agg.emml	Chained RAQL queries to compute Monthly, Quarterly and Yearly Aggregations
intersect.emml	RAQL intersect operator to find intersection of 2 datasets
join-raql.emml	Various RAQL Join queries to illustrate Join of two Stream Sources; two DOM Sources; and a Stream Source with DOM source.
join2-raql.emml	RAQL Join cross diverse data formats - joins XML and CSV datasources. RAQL Join on columns transformed using functions and cast functions.
joinBigMemory.emml	RAQL Joins 2 BigMemory cache sources; Also illustrates join between BigMemory and Stream/DOM sources.
kmeans1.emml	RAQL query for K-Means clustering using Euclidean and Manhattan measures
kmeans2.emml	RAQL K-Means centroids computation using Euclidean and SquaredEuclidean measures
legislators.emml	Diverse RAQL queries to illustrate usage of - select columns; limit results; distinct records; filter expressions; like search; group aggregations; order by.

Sample name	Short Description
linear_regression_1.emml	Use of RAQL Linear Regression analytic function
load-decl-cache.emml	Loads 'Legislators' sample data into cache; for subsequent analysis using assorted queries from query-decl-cache.emml
load-stocks-cache.emml	Loads 'Stocks' data into cache; for subsequent analysis using assorted queries from query-stock-cache.emml
math1.emml	Use of basic arithmetic operators in RAQL queries
mfgplants-cache.emml	Loads 'mfgPlants.csv' CSV dataset into cache; for subsequent analysis using query-mfgplants-cache.emml
mfgplants.emml	Various RAQL queries (Group by, Filter, Sub-query) on CSV dataset
mysql-raql.emml	Use of SQL with RAQL. SQL does base query and sends results to RAQL for windows aggregation.
parallel-query1.emml	ForEach parallel statement to perform RAQL queries in parallel and then combine results using <fuse/> statement
percent-change-subquery.emml	RAQL query to calculate Volume Mean Percentage change over a certain threshold value, across each quarter. This is done using Sub Query.
percent-change.emml	RAQL query to calculate Volume Mean Percentage change over a certain threshold value, across each quarter. This is done using Chained Query.

Sample name	Short Description
query-decl-cache.emml	see load-decl-cache.emml
query-mfgplants-cache.emml	see load-mfgplants-cache.emml
query-stocks-cache.emml	see load-stocks-cache.emml
schema2.emml	EMML In-line Schema definition for RAQL datasources
snapshot.emml	QueryMashZone NextGenSnapshot data using RAQL
snapshot2.emml	Use of alternate pattern to queryMashZone NextGenSnapshots using RAQL
stocks-group-avg.emml	Group avg aggregation of stocks by month, quarter, year
stocks-group-count.emml	Group count aggregation stocks by month, quarter , year
stocks-group-max.emml	Group max aggregation of stocks by month, quarter , year
stocks-group-min.emml	Group min aggregation of stocks by month, quarter , year
stocks-group-stddev.emml	Group stddev of stocks by month, quarter, year
stocks-group-sum.emml	Group sum aggregation by month, quarter, year
stocks-group-variance.emml	Group variance of stocks by month, quarter, year
stocks-price-chg.emml	Using RAQL chained queries compute min and max opening price change between weeks.

Sample name	Short Description
stocks-running-avg.emml	Running avg of stocks by month, quarter, year
stocks-running-count.emml	Running count by month, quarter, year
stocks-running-max.emml	Running max of stocks by month, quarter, year
stocks-running-min.emml	Running min of stocks by month, quarter, year
stocks-running-stddev.emml	Running stddev of stocks by month, quarter, year
stocks-running-sum.emml	Running sum of stocks by month, quarter, year
stocks-running-variance.emml	Running variance of stocks by month, quarter, year
subquery1.emml	RAQL query to calculate mean percentage chagne in stock volume over 5% between quarters; implemented using RAQL Sub Query
udf-sample.emml	RAQL query using User-defined Functions
union.emml	RAQL union operator illustrates combining datasets with following properties ■ 2 stream sources ■ 2 DOM sources ■ XML and CSV source formats
window-functions-using-cache.emml	Query against BigMemory cache using Window functions - rank, dense_rank, lag, lead, first_value, last_value, row_number

Sample name	Short Description
window-functions.emml	Query against file datasource using Window functions - rank, dense_rank, lag, lead, first_value, last_value, row_number

9 Views for Mashups and Mashables

- Built-in MashZone NextGen Views 901
- Creating Pluggable Views or Libraries 1045

Views define the format and layout for data from mashables and mashups. Views may also require specific types of information. Charts typically work with numeric data, maps require location data and tabular views accept almost any kind of data.

Views also define which devices support their layout: desktop browsers, mobile phones and/or mobile tablets. See "About Desktop and Mobile View Compatibility" on page 903 for more information.

MashZone NextGen bundles a set of common views, known as the *built-in views*, that you can use to work with mashable or mashup data or use to create apps. New in this release, MashZone NextGen also now includes *real-time* views for Event mashables.

You can add multiple views to use with a specific mashable or mashup. See "Add Views to Mashables and Mashups" on page 901 for instructions. You can then use these views can in basic apps that you or other users create from that mashable or mashup. Or they can be added directly to workspaces. See "Create a Basic App" on page 1093 for more information on creating basic apps. See "Create Workspace Apps with Mashboard" on page 1116 for more information on creating workspaces.

MashZone NextGen developers and administrators can also:

- Customize the views you create for mashables and mashups. This actually creates basic apps which they customize. See "Customize a Basic App or View" on page 1198 for instructions and examples.
- Create additional, pluggable views that are added to the list of built-in views. You can then use them to add views to mashables or mashups. See "Creating Pluggable Views or Libraries" on page 1045 for instructions and examples.

The following built-in MashZone NextGen views are all compatible with desktop browsers. Views that are also mobile compatible, have additional icons. See "About Desktop and Mobile View Compatibility" on page 903 for more information.

Grids	■ Feed Reader : displays syndicated feeds in either RSS or Atom formats in a table. ■ Grid and KPIs View : displays data in a table of any size providing a variety of formats for data including KPI rules to display status indicators. This view automatically paginates data and can group rows based on the unique values in one column. ■ Record Details View : shows detailed data for a single row of repeating data in the results in a two-column table. The first column shows field labels and the second shows values. You may 'page' through each row of values, as needed.
Maps	■ Geo Map : displays one numeric statistic for one or more countries within a map of a specific region or of the entire world. Numeric values display as a gradient of a color, from light to dark corresponding to lowest to highest values.

	■ Google Map : shows one or more repeating items in your data as locations in a map with an overlay of roads and streets. Users can choose different *map types* and can control placement and zoom. **Note:** This view requires Internet access. ■ Intensity Map : displays one or several numeric statistics for one or more countries within a map of a specific region or of the entire world. Numeric values display as a gradient of a color, from light to dark corresponding to lowest to highest values. Each statistic displays in a separate tab.
Line and Area Charts	■ Area Chart : displays one or more numeric dimensions for a set of records as lines between data points for each record and the area underneath each line filled with color. ■ Kagi Chart : displays stepped lines indicating the change in trend for one numeric dimension compared to thresholds, such as changes in supply and demand based on price with thresholds of previous high and low. ■ Line Chart : displays several numeric dimensions for a set of records. Each dimension draws angled straight lines that illustrate the change between data points and the lines emphasize the differences between records for the same data point. ■ Sparkline : displays the fluctation of a single data set over time or categories. Data should have small fluctuations and be appropriate to render as a line, such as the fluctuations of a stock price. ■ Spline Area Chart : displays one or more numeric dimensions for a set of records as curved lines with the area underneath each curve filled with color. This is similar to area charts, but the fitted curves are appropriate for continuous data. ■ Spline Line Chart : displays several numeric dimensions for a set of records as curved lines that illustrate the trend or change between data points. This is similar to line charts, but the fitted curves are appropriate for continuous data. ■ Step Line : displays lines that are can be vertical or horizontal only, resulting in *steps* that highlight changes in magnitude. Step lines are can be more appropriate for discrete data or for data where specific thresholds should be visible.

Bar, Column and Rectangle Charts	■ Bar Chart : displays numeric data points as horizontal bars for a set of records. For vertical "bars", use the Column chart instead. Bar can be rendered in two dimensions or three. Bars can also be stacked. ■ Candlestick Chart : displays price and optionally volume information for stocks or commodities in a grid of *candles* and optional column chart. ■ Column Chart : displays numeric data points as vertical bars for a set of records. For horizontal "bars", use the bar chart instead. Columns can be rendered in two dimensions or three. Columns can also be stacked. ■ Gantt Chart tracks the progress of tasks across a timeline. ■ Marimekko Chart : displays multiple numeric dimensions as stacked columns allowing easy comparison across different categories. Columns may also display dimensions as percentages. ■ Pareto Chart : a combination of a column chart and a line chart, pareto charts order data in an 80/20 display to allow easy identification of key causes or factors. ■ Sparkline Column : displays a single data metric over time or categories as a set of vertical bars. ■ Sparkline win_loss : displays a single data metric across time or categories with respect to an origin line. One typical use is to dsiplay data points as win/loss/draw or profit/loss/break-even.
Circular and Miscellaneous Charts...	■ Bubble Chart : displays numeric data points positioned on a grid and also indicate the importance or scale of each data point by the size of the bubble. Bubble charts are a variation of scatter charts. ■ "Doughnut Chart" on page 930 : displays numeric data points for a single record or a single data point for multiple records as percentages of a whole. The lack of a center gives a different emphasis to the data. See also Pie chart. ■ Funnel Chart : displays a single, progressively smaller numeric data point for a set of records. Data decrease down to the final point of the funnel. Both the decreasing radius of the funnel and the overall depth of each slice indicate the decreasing size of the data. See also Pyramid chart.

	■ Gauge Set : see following row. ■ Pie Chart : displays numeric data points for a single record or a single data point for multiple records as percentages of a whole. The percentage is rendered as a "slice of pie" covering an arc within a circle. See also Doughnut chart. ■ Pyramid Chart : displays a single, progressively smaller data point for a set of records. Data decreases up the pyramid to the final point. Both the decreasing circumference of the pyramid and the overall depth of each slice indicate the decreasing size of the data. See also Funnel chart. ■ "Radar Chart" on page 992 : display irregular polygons plotted from a shared center with each polygon point representing the magnitude of a dimension of one item. Radar charts facilitate easy comparison of multiple dimensions for dataset rows. This is also sometimes called a spider chart. ■ Scatter Chart : displays a series numeric data points as discrete points. Scatter charts allow users to determine a pattern, if any, and the scope or distribution of the data. See also Line and Area charts.
Gauges	■ Dial Gauge : angular gauges, or *dials*, display a *single* numeric data point, or a set of data points, in relation to a scale rendered in an arc. With multiple data points, each value appears in a single gauge and users can page to the gauge for the next value. ■ Bulb Gauge : shows a single numeric data point using the size, color and label or percentage to indicate the status of the data in relationship to a scale. Bulb graphs can also handle sets of numeric values. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value. ■ Bullet Gauge : display a single numeric data point as a bar in relation to a scale and a *target* on that scale. Bullet graphs are similar to bar or column charts and can be either horizontal or vertical. Bullet graphs can also handle sets of numeric values. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value. ■ Cylinder Gauge : displays a single numeric data value, or a set of single values, in relation to a scale rendered as a cylinder. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value. ■ LED Gauge : displays a single numeric data point, or a set of data points. in relation to a scale that is rendered as

	LED lights. This scale can also progress through color ranges to provide additional status information. The scale can be either horizontal or vertical. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value. ■ Linear Gauge : displays a single numeric data point, or a set of numeric data points, in relation to a scale that renders as a horizontal bar. The bar can also progress through color changes to provide additional status information. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value. ■ Thermometer Gauge : displays a single numeric data point, or a set of data points, in relation to a scale that is rendered as a vertical thermometer. This can be literal temperature data or simply a scale to indicate importance, progress and so on. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value.
Real-Time Charts	Real-time charts provide common chart views *only* for event mashables. See "What are real-time charts?" on page 74 for more information. Available real-time charts include: ■ Real-Time Line Chart : displays one or more numeric dimensions as lines for a set of events that either: ■ Dynamically updates a growing time line. ■ Dynamically updates lines for one category.
Custom	MashZone NextGen developers can also create custom views for a specific mashables or mashup using the Template View. MashZone NextGen developers can also create pluggable views and add them to the MashZone NextGen View Gallery to allow users to add pluggable views to many mashables or mashups. See "Creating Pluggable Views or Libraries" on page 1045 for instructions and examples.

Add Views to Mashables and Mashups

Views define how mashup and mashable response data is displayed, the layout and format of the data, for one or more devices: desktop browsers, mobile phones and mobile tablets. Views can also define additional behaviors, such as navigation to multiple pages of rows within the response.

MashZone NextGen automatically defines a Tree view and, if possible, a Grid view for mashups and mashables. Tree view displays the structure of the response, while Grid provides a table view of the data.

Mashups and mashables can have many different views. You must have appropriate permissions in MashZone NextGen to add views to a mashup or mashable.

New in this release, MashZone NextGen includes *real-time views* for Event mashables. You can only add real-time views directly to workspaces in Mashboard. For all other built-in MashZone NextGen views:

1. Open the mashup or mashable, if needed, and run the mashup or mashable to preview response data. See "Run and Preview Mashable/Mashup Data" on page 177 for instructions.

   If the mashup or mashable is new, the preview displays data in the Grid View or Tree View.

2. Click **View >** **Add View**.

   A gallery of views opens displaying the views bundled in MashZone NextGen, and optionally pluggable views that have been added by a MashZone NextGen developer or administrator.

   Note: All built-in MashZone NextGen views are compatible with desktop devices and browsers, but some are not compatible with mobile devices. Device compatibility for pluggable views may also vary.

   The View Gallery shows the device compatibility for a view using these icons:

   - **Desktop compatible** icon
   - **Mobile compatible** icons: or .

3. Select the view you want to add for the devices that you want to support.
4. Follow the steps in the Create View window to complete configuration for the view:
   - In **Configure Data**, configure the data that you want to see in the view.
   - Optionally, change the appearance of the view in **Customize Appearance**.

The options to configure the data and optionally configure the appearance of the view are different for each view. For specific information for the MashZone NextGen built-in views, see:

"Area Chart" on page 912	"Geo Map" on page 941	"Radar Chart" on page 992
"Bar Chart" on page 914	"Google Map" on page 944	"Record Details View" on page 988
"Bubble Chart" on page 916	"Grid and KPIs View" on page 946	"Scatter Chart" on page 994
"Candlestick Chart" on page 924	"Intensity Map" on page 966	"Sparkline" on page 995
"Column Chart" on page 926	"Kagi Chart" on page 968	"Sparkline Column" on page 997
"Doughnut Chart" on page 930	"Line Chart" on page 972	"Sparkline win_loss" on page 999
"Feed Reader" on page 933	"Marimekko Chart" on page 982	"Spline Area Chart" on page 1001
"Funnel Chart" on page 934	"Pareto Chart" on page 984	"Spline Line Chart" on page 1002
"Gantt Chart" on page 936	"Pie Chart" on page 986	"Step Line" on page 1004

"Gauges" on page 941	"Pyramid Chart" on page 989	"Template View" on page 1006

- Use **Preview** to review the effect of your changes in all the devices that this view supports. Use the toolbar to switch the preview to:

5. Once you are satisfied with the view, move to **Done** and:
   a. Enter a name for this view.

      View names must be unique for the mashable or mashup that they are assigned to. They can contain characters from the character sets supported by the MashZone NextGen Repository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ -* ' .`

      Tip: View names can appear as the title of the view in apps that include multiple views. It is a good practice to make view names clear and easy to understand.

   b. Optionally, enter a brief description for this view.
   c. Click **Save**.

About Desktop and Mobile View Compatibility

The View Gallery shows the device compatibility for both the built-in MashZone NextGen views and pluggable views, if any, with these icons:

- **Desktop compatible**. All of the built-in MashZone NextGen views are compatible with desktop browsers.
- **Mobile phone compatible**. Most, but not all of the built-in MashZone NextGen views are compatible with mobile phones.
- **Mobile tablet compatible**. Most, but not all of the built-in MashZone NextGen views are compatible with mobile tablets.

Mobile compatibility means that the view has been optimized for mobile devices. Some view configuration properties may not apply to mobile views. Mobile compatibile views also recognize and respond, where appropriate, to common gestures.

Note: Because workspace apps are not currently compatible with mobile devices, the publish events in views that are used in workspaces to allow apps to work together are not applicable in mobile devices.

Manage Views

For mashables and mashups that you create, you, or MashZone NextGen administrators, manage what views are available to be used in apps and what the default view should be in when users preview the mashable or mashup in the artifact page. You use the **View** menu and the Views selection list:

- **View > Make Default** flags the current view you have selected as the default view to show results when users run this mashable or mashup in the artifact page.
- **View > Edit** allows you to edit the configuration for this view.
- **View > Delete** permanently removes the view for this mashable or mashup.

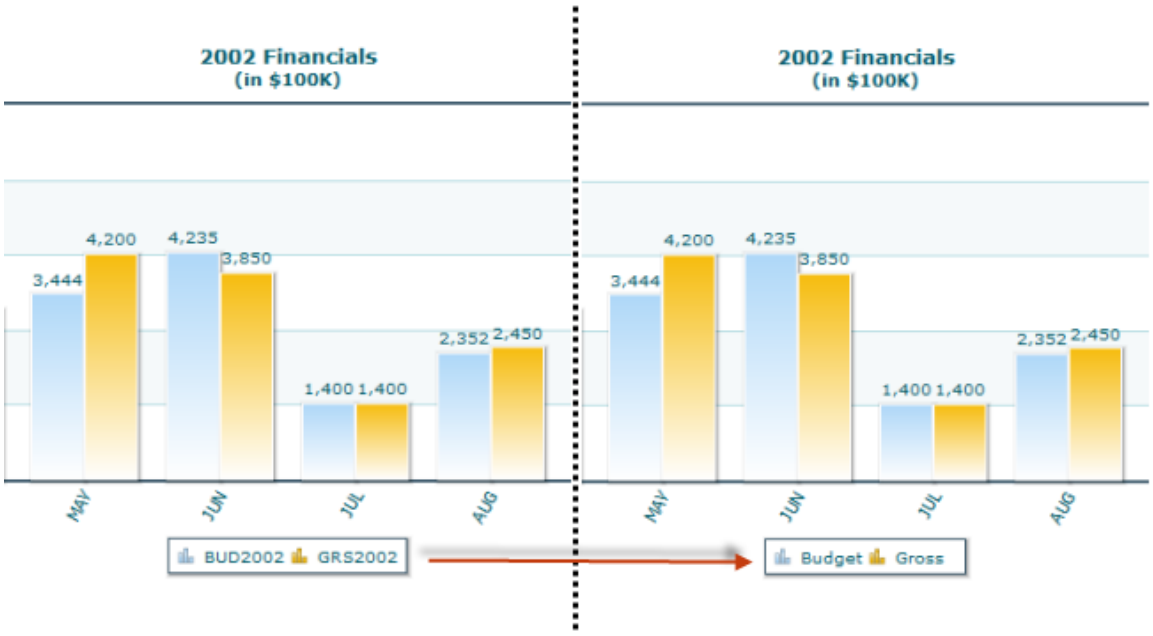
Common Chart Techniques

There are several techniques that you can use in many charts, including:

- Change Series Labels
- Change the Series Plot to Column, Line or Area
- Add a Target Line
- Include Dynamic Content in Captions and Labels

Change Series Labels

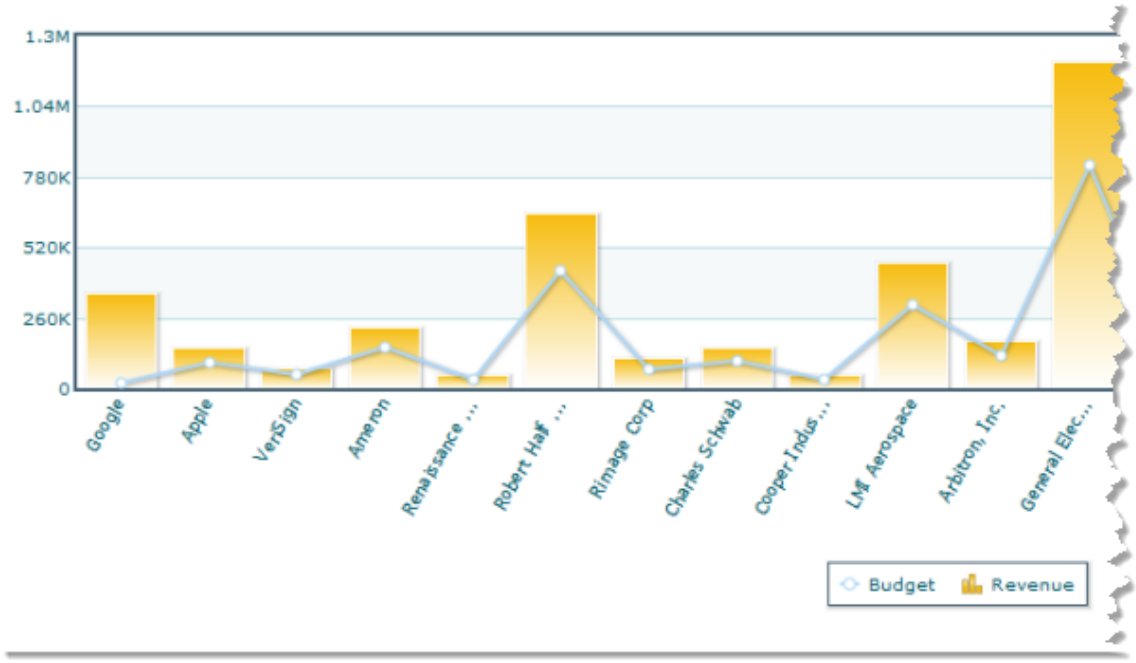
In some cases, the name for columns are abbreviated or difficult to understand. You can change the label to show in your chart for columns used as *series* or the *X column*.



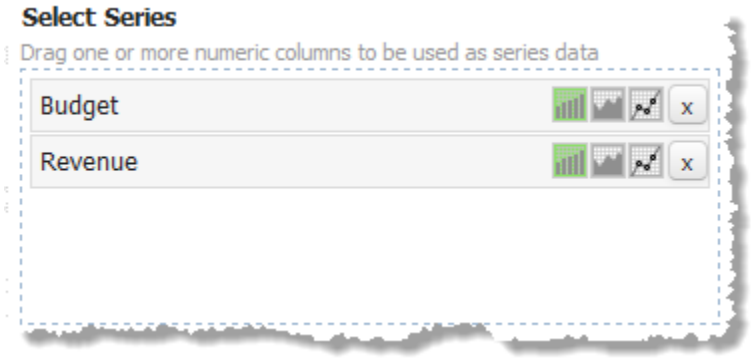
To change the label for a column that you have assigned as a **Series** or **X Column** for this chart, double-click the column name and enter the label you want to use.

Change the Series Plot to Column, Line or Area

With some charts that can have multiple numeric series, you can choose to plot the data for each series in a different form, such as this combination Line and Column chart:

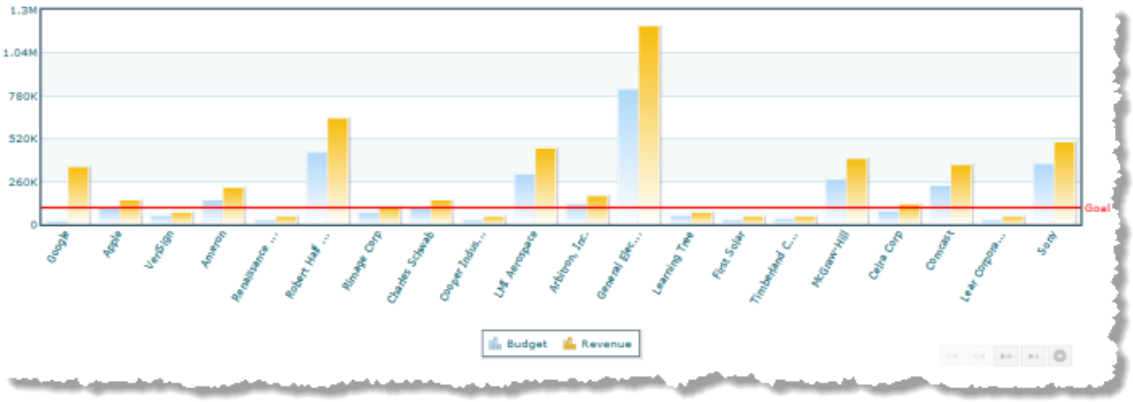


Simply click the column, area or lin button for that series:



Add a Target Line

With some types of charts, you can add a single, horizontal line to the chart to indicate a goal, theshold or specific target, such as this example:



1. Enter a **Label** for the target line, if desired.
2. Enter the **Value** for the target.

Include Dynamic Content in Captions and Labels

You can include dynamic information, such as the name of the currently selected customer, in the following chart captions or labels:

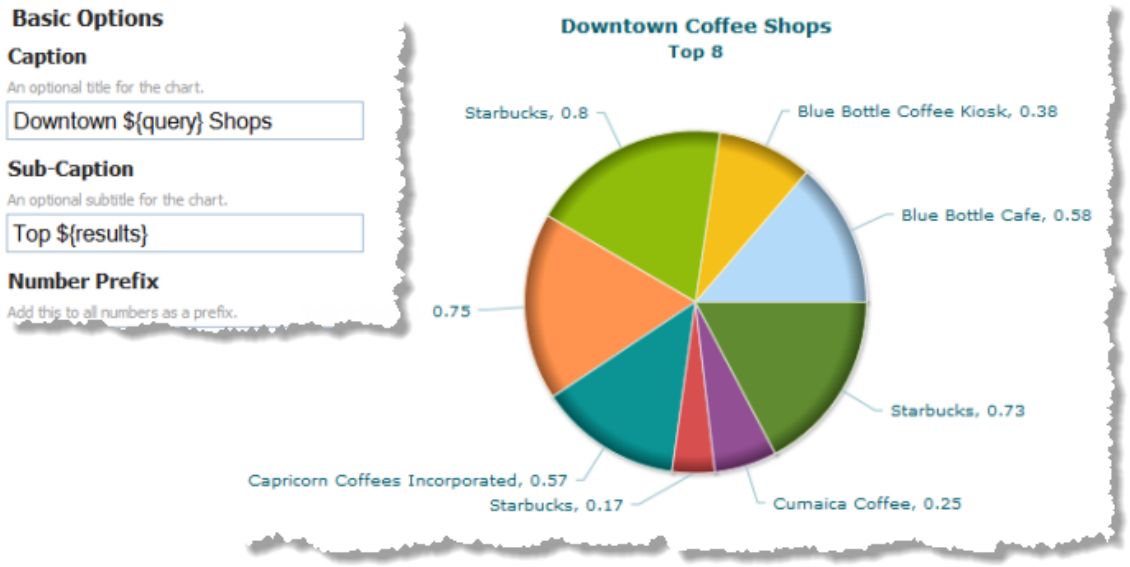
- Caption
- Sub-Caption
- X Axis Label

  If a chart supports multiple axes, all axes labels can contain dynamic content
- Y Axis Label

  If a chart supports multiple axes, all axes labels can contain dynamic content

- Number Prefix
- Number Suffix
- Legend Caption

This example chart has dynamic content in both the caption and sub-caption:



To add dynamic content to chart captions and labels:

- Either:
  - The mashup or mashable for the chart has input parameters that provide the dynamic content.
  - You define additional properties for the chart that will receive dynamic content from other apps or views in a workspace app using wiring.
  - Or both.
- You add a *template* in the caption or label property, as you configure the view, in the form:

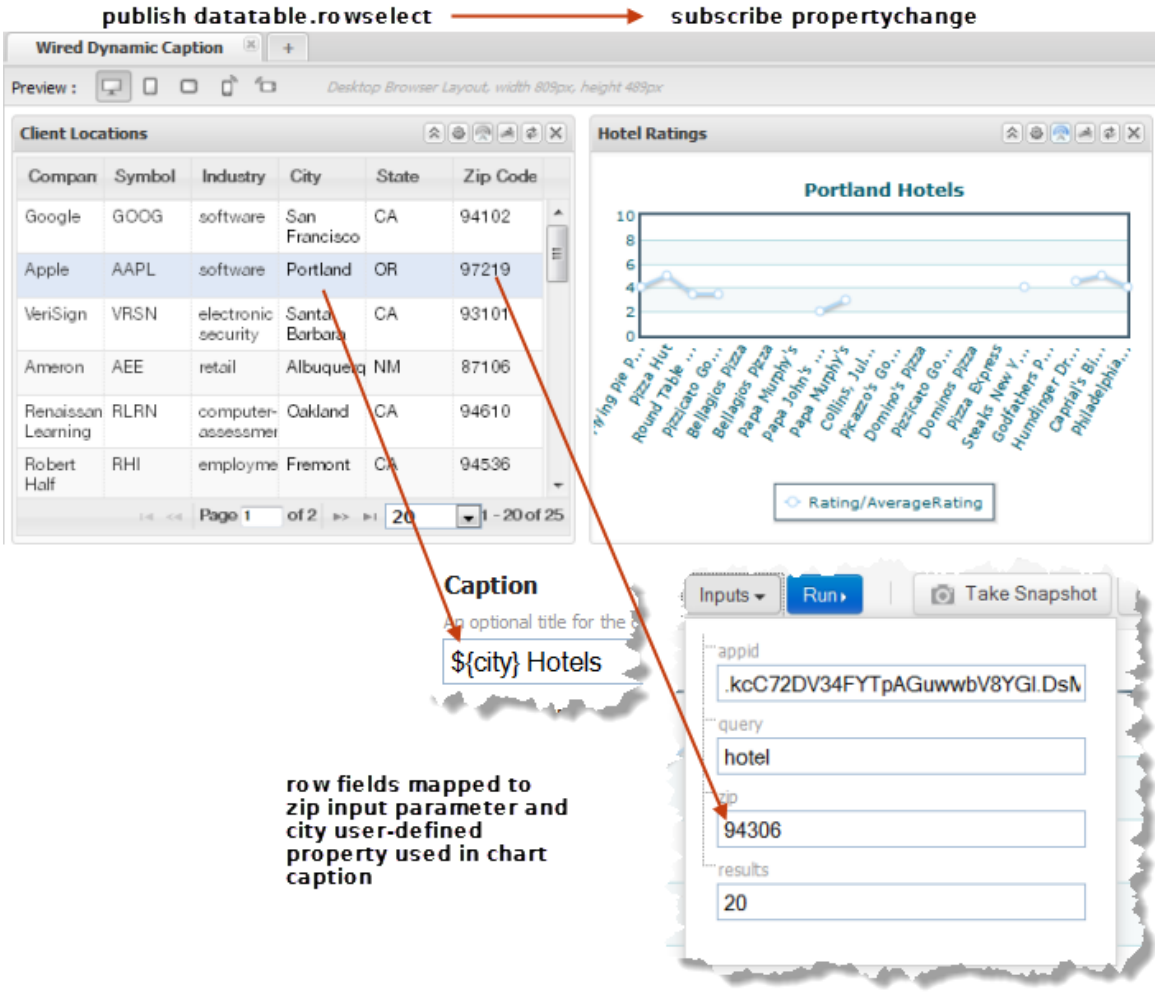
  `${`*input-parameter-or-user-defined-property-name* `}`

  The chart shown above is an example of dynamic caption content using input parameters. The mashable in this example has a `query` and a `results` input parameter.
- If an input parameter or user-defined property is optional and is not set, you can provide a default value to add to captions and labels using a template in this form:

  `${`*input-parameter-name=default-value* `}`

  For example, `${results=10}` would provide a default number for the sub-caption in the previous example.

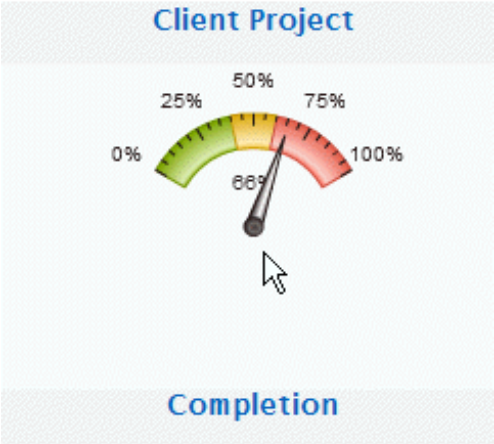
- If you use user-defined properties in a caption, add the view or an app that uses the view to a workspace app in Mashboard and wire the property to fields in a published message from another view or app in the workspace. For example:



Dial Gauge

Dials, or *angular gauges*, display a *single* numeric data point in relation to a scale rendered in an arc. As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with one type of gauge or a mix of gauge types.

Note: Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with an angular gauge

1. Add a Gauge to This View
2. Configure a Dial Gauge
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view and once you are satisfied, save the view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

   Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.

2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

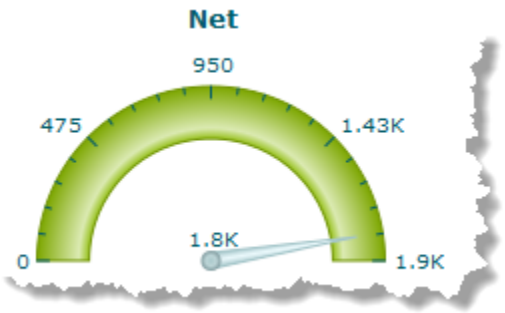
   Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.

3. Choose the type of gauge you want and complete configuration properties.

Configure a Dial Gauge

1. If needed, select the Series column for the gauge and change the **Gauge Type** to `Dial`.
2. Define the scale and color range for this gauge in the **Color Ranges**.

   The scale defaults to 0-100 or a maximum value based on your data, with a single color range:



   Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest values that you want to show.

   You define the number of colors that appear in the scale by how many intermediate numbers you include in the scale. With no intermediate numbers, the scale is one color. Add an intermediate number to change the scale color at that point.

   Note: You can also change the colors used in the scale in the **Customize View** step of the view wizard.

   You can add numbers or remove numbers from the scale.

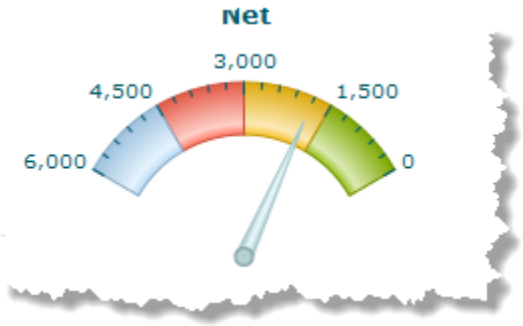
- To add a new number to the scale, enter a number and click **Add Value**.
- To remove a number, click on the number in the scale.

3. Enter the title for *this* specific gauge as the **Title**.
4. Define the arc and direction for the *dial* for this gauge in the **Start Angle** and **End Angle** properties.

   By default, the arc of the scale is a semi-circle, starting from 0 up to 180 degrees. You can decrease or increase the arc by defining the degrees for the start and end points.

   The default direction for the dial is clock-wise. To change the direction use a higher degree as the start point and a lower degree as the end point.

   For example, with a start of `100` and an end of `0` the dial would look something like this:



5. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

   c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

   d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

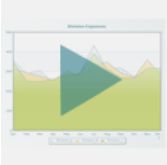
   e. Click **Add Color** to add this color to the list of colors used in the dial

   f. Add a color for each range of numbers you defined for the dial.

      If needed, simply click on a color in the list to delete that color.

6. Preview the gauge, as needed.

Area Chart



Area charts display numeric statistics for repeating data as a filled area below a line in a coordinate grid defined by X and Y axes. Areas can also be stacked. This view supports desktop and mobile devices.

Tip: Data can be somewhat obscured in an unstacked area chart when one data point is lower than the corresponding point for the top-most area. Consider a Line Chart or a Zoom Line Chart as an alternative.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what is on each tick of the X axis. This is the *category* field. In the example shown above, the Month field is the category. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines an area filled with one color. In the example shown above, there are three series for Division_a, Division_b and Division_c.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Area Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

> **Tip:** With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a line and filled area in a different color to the chart.

   > **Tip:** For best results, limit the number of series (the number of lines per category) to 12 or less. For stacked charts, limit the number of series to eight or less.
   >
   > It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, Change the Series Plot to Column, Line or Area or Change Series Labels to make the chart easier to understand.

4. Optionally, change either of these properties:
   - **Stack Series?** to stack the areas of each series across all categories.
   - **Chart Scroll** set this to yes to allow the categories to spill over to additional pages when you have larger numbers of categories.
5. If desired, Add a Target Line to the chart.
6. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

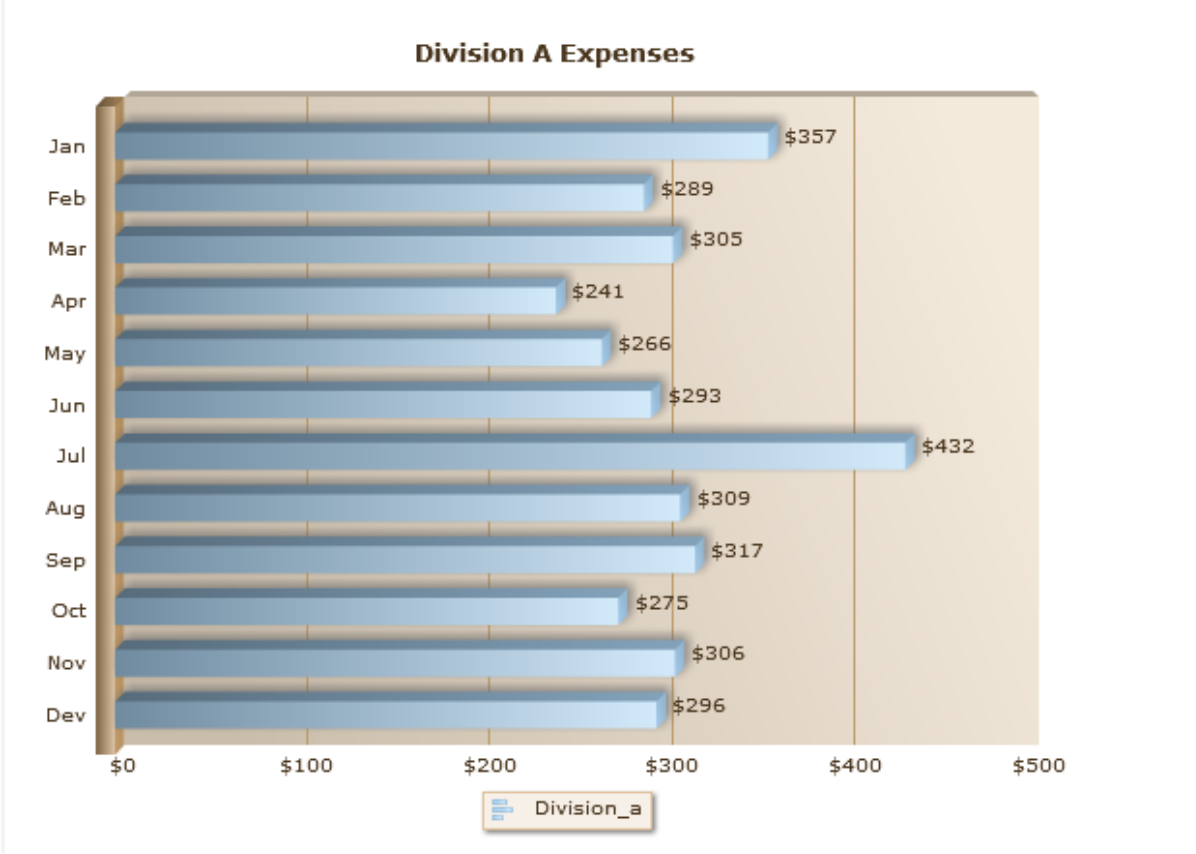
     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
7. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
8. Preview the chart at any time.
9. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Bar Chart

Bar charts display numeric statistics as horizontal bars for a set of records. Bar chart views support desktop and mobile devices.

Bars can be rendered in two dimensions or three. Bars can also be stacked. For vertical "bars", use the Column Chart instead. For a reverse relationship between records and data points, see "Sparkline Column" on page 997.



There are two common patterns for data in bar and column charts:

- Numerical data distributed by a frequency, such as time intervals.

  This is some times called a *histogram*. Although time is a common distribution, it isn't required. Any numeric distribution works.

- Numeric data sorted into categories, such as expense categories.

  With categories, the order in which categories are presented is defined by the order it appears within results.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what is on each tick of the Y axis. This is the *category* field. In the example shown above, the Month field is the category. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines the bars that are graphed with one color. In the example shown above, there is one series for Division_a.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Bar Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

   Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a set of bars in a different color to the chart.

   Tip: For best results, limit the number of series (the number of bars or columns per interval/category) to 12 or less. With stacked bars, limit the number of series to eight or less for best results.

   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, Change Series Labels to make the chart easier to understand.

4. Optionally, change any of these properties:
   - **Stack Series?** to stack the series in one column for each category.

- **Chart Visualization** to set this as two dimensional (2D) or three dimensional (3D).

5. If desired, Add a Target Line to the chart.
6. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
7. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
8. Preview the chart at any time.
9. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

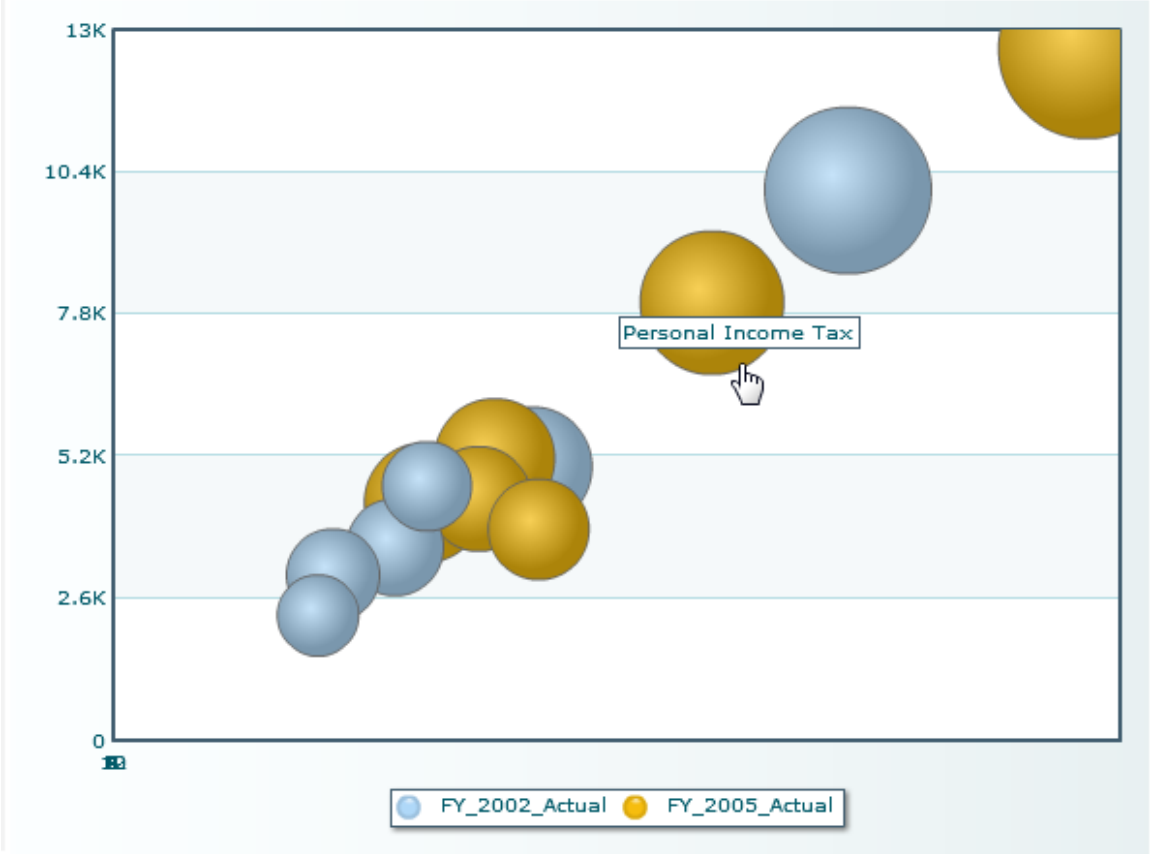
Bubble Chart

Bubble charts display numeric statistics plotted as discrete points on a grid and also indicate the importance or scale of each data point by the size of the bubble. Bubble chart views support desktop and mobile devices.

Bubble charts allow users to determine a pattern, if any, in the data and the scope or distribution of the data. Bubble charts are a variation of Scatter Charts.

> **Tip:** For best results with bubble charts, limit the number of repeating items to be plotted to 30 or less.

You can plot more than one set of points in a bubble chart. See "Characteristics" on page 917 and "Configure This View" on page 917 for more information.



Characteristics

Data Requirements	A set of repeating items with: ■ At least three fields with numeric data to use as the X and Y coordinates of the points and the radius of the bubbles to plot in the chart. ■ An alphanumeric field to use as a tooltip for each point.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the Bubble Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Complete one **Series**:
   a. Drag and drop the title of columns with numeric data to both the **X Column** and **Y Column** fields.

      If needed, Change Series Labels for the X column to make the chart easier to understand.

   b. Drag and drop the title of a third column with numeric data in the **Radius** field to use to determine the importance and the radius of each bubble.

   c. Drag and drop the title of a column with text or numeric data into the **Tooltip** field.
3. To plot additional points in the bubble chart, click **Add series columns**. This collapses the fields for the previous Series and opens fields for another series. Complete the fields (see the previous step for instructions) for this new series.

   Tip: For best results, the numeric data for all of the series that you include in a bubble chart should fall within a similar range of values. One series with very large or very small values can skew the scale of the chart, making it difficult to see or understand other data.
4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.

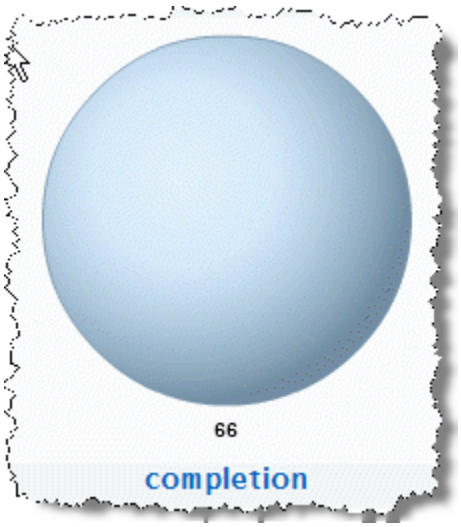
   b. Give the view a name, and optional description.

   c. Click **Save**.

Bulb Gauge

The bulb gauge shows a single numeric data point using the size, color and label or percentage to indicate the status of the data in relationship to a scale. As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with one type of gauge or a mix of gauge types.

Note: Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with a bulb gauge

1. Add a Gauge to This View

2. Configure the Bulb Gauge
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view. Once your are satisfied, save this view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

   Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.
2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

   Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.
3. Choose the type of gauge you want and complete configuration properties.

Configure the Bulb Gauge

1. If needed, select the Series column for the gauge and change the **Gauge Type** to `Bulb`.
2. Define the scale and color range for this gauge in **Color Ranges**.

   The scale defaults to 0-100 or a number close to the maximum value for the selected column, with a single color.

   Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest numbers that you want to show.

   You define the number of colors that appear in the scale by how many intermediate numbers you include in the scale. With no intermediate numbers, the scale is one color. Add an intermediate number to change the scale color at that point.

   Note: For bulb gauges, the bulb color is set by where the bulb value is on the scale. But *only* one color shows for a given bulb.

You can add numbers or remove numbers from the scale.

- To add a new number to the scale, enter a number and click **Add Value**.
- To remove a number, click on the number in the scale.

3. Enter the title for *this* specific gauge as the **Title**.
4. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.
5. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

   c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

   d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

   e. Click **Add Color** to add this color to the list of colors used in the dial

   f. Add a color for each range of numbers you defined for the dial.

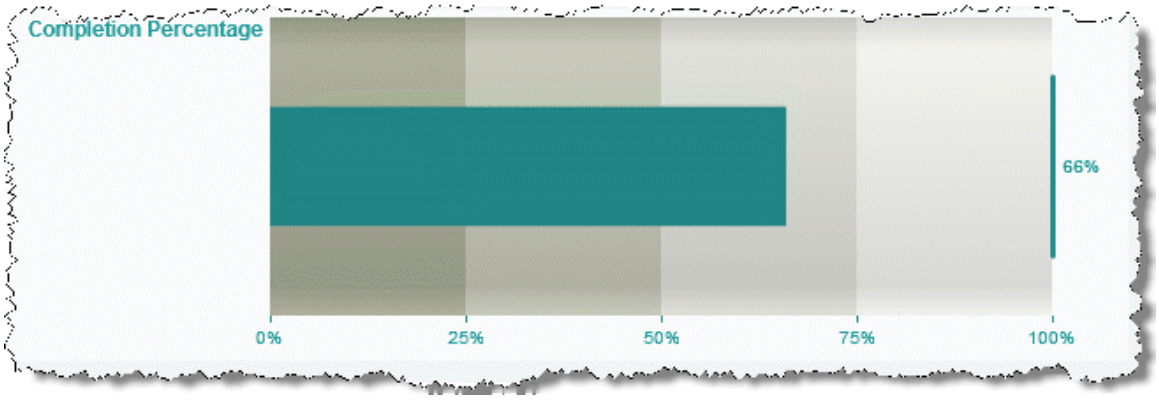
      If needed, simply click on a color in the list to delete that color.
6. Preview the gauge, as needed.

Bullet Gauge

Bullet gauges display a single numeric data point as a bar in relation to a scale and a *target* on that scale. Bullet graphs are similar to bar or column charts and can be either horizontal or vertical.

Note: Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.

As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with a mix of gauge types.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with a bullet gauge:

1. Add a Gauge to This View.
2. Configure a Bullet Gauge.
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view and once you are satisfied, save the view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.

2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.

3. Choose the type of gauge you want and complete configuration properties.

Configure a Bullet Gauge

1. If needed, select the Series column for the gauge and change the **Gauge Type** to `Bullet`.
2. Define the scale for this gauge in the **Color Ranges**.

   The scale defaults to 0-100, or a maximum number close to the maximum value for this column. Unlike some other types of gauges, intermediate numbers in the scale do *not* affect the fill color for the bullet graph bar, but instead affect the fill of the background scale.

   Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest numbers that you want to show.

   You can add numbers or remove numbers from the scale.

   - To add a new number to the scale, enter a number and click **Add Value**.
   - To remove a number, click on the number in the scale.
3. Use the **Chart Orientation** property to set the direction of the bar for this bullet graph.
4. If you want the gauge to indicate a target or desired goal, enter the **Target** number.

   The target is shown in the bullet graph as a line (see above).
5. Enter the title for *this* specific gauge as the **Title**.
6. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.
7. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

e. Click **Add Color** to add this color to the list of colors used in the dial

f. Add a color for each range of numbers you defined for the dial.

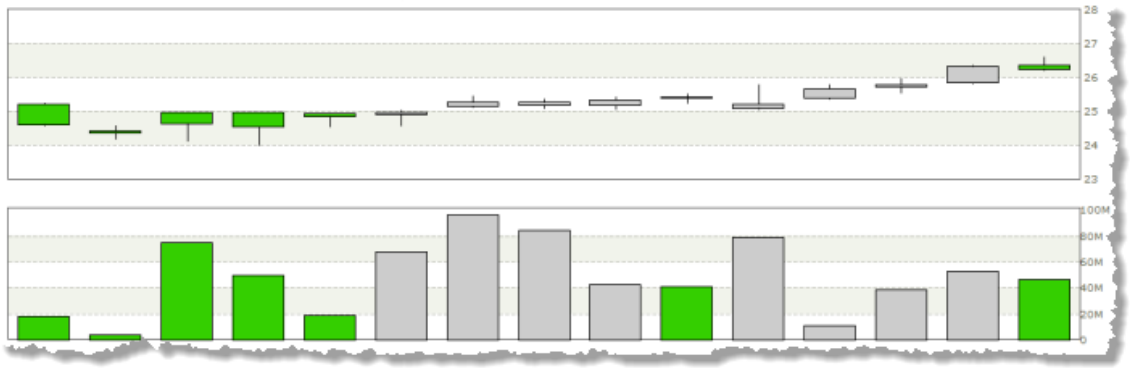
   If needed, simply click on a color in the list to delete that color.

8. Preview the gauge, as needed.

Candlestick Chart

The candelstick chart summarizes two separate pairs of numeric measures, plus an optional separate measure in an easily learned format for quick comparisons. They are most commonly used to plot stock or commodity prices, and optionally volume, along with the current trend (bear or bull) of the price.

Candlesticks combine a column overlayed with a vertical line that looks something like a candle and wick. The *candle* shows the start and end values (open and close) while the wick shows high and low values. Color also indicates the overall trend . It can also include a separate column chart indicating volume.



Characteristics

Data Requirements	A set of repeating items with:
	■ Four numeric fields that represent different measures of the same dimension (such as price) in two related pairs:
	  ■ Open and Close
	  ■ High and Low
	■ Optionally an additional numeric field that is plotted in a separate bar graph for the Volume. Typically, the range

	of values for this this field can be very different than the range for the required measures. ■ Optionally, a field with a label identifying the stock, commodity or other entity being tracked.
Mobile Views	Yes

Configure This View

Choose the Candlestick Chart and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the columns with numeric data and drop them in the following fields:
   - **Opening price**
   - **Highest price**
   - **Lowest price**
   - **Closing price**
3. Optionally, drag the title of a numeric and drop it in the **Volume** field.

   If you provide volume data, the chart will include a separate bar chart plotting volumne only. If you omit a volume, only the candlestick chart is plotted.
4. Optionally, drag a text column to provide labels in the X axis and drop this in the **Categories** field.
5. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
6. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
7. Preview the chart at any time.
8. Once you are satisfied, save the view:
   a. Click **Finish**.

b. Give the view a name, and optional description.

c. Click **Save**.

Column Chart

Column charts display numeric data points as vertical bars for a set of records. Columns can be rendered in two dimensions or three. Columns can also be stacked. For horizontal "bars", use the "Bar Chart" on page 914 instead. For a reverse relationship between records and data points, see "Sparkline Column" on page 997.

There are two common patterns for data in bar and column charts:

- Numerical data distributed by a frequency, such as time intervals.

  This is some times called a *histogram*. Although time is a common distribution, it isn't required. Any numeric distribution works.

- Numeric data sorted into categories, such as expense categories.

  With categories, the order in which categories are presented is defined by the order it appears within results.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what is on each tick of the X axis. This is the *category* field. In the example shown above, the Month field is the category. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines the columns that are graphed with one color. In the example shown above, there are three series for Division_a, Division_b and Division_c.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Column Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

   Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a set of columns in a different color to the chart.

   Tip: For best results, limit the number of series (the number of bars or columns per category) to 12 or less. With stacked columns, limit the number of series to eight or less for best results.

   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, "Change the Series Plot to Column, Line or Area" on page 905 or "Change Series Labels" on page 904 to make the chart easier to understand.

4. Optionally, change any of these properties:
   - **Stack Series?** to stack the series in one column for each category.
   - **Chart Visualization** to set this as two dimensional (2D) or three dimensional (3D).
   - **Chart Scroll** set this to yes to allow the categories to spill over to additional pages when you have larger numbers of categories.
5. If desired, Add a Target Line to the chart.
6. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.

7. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.

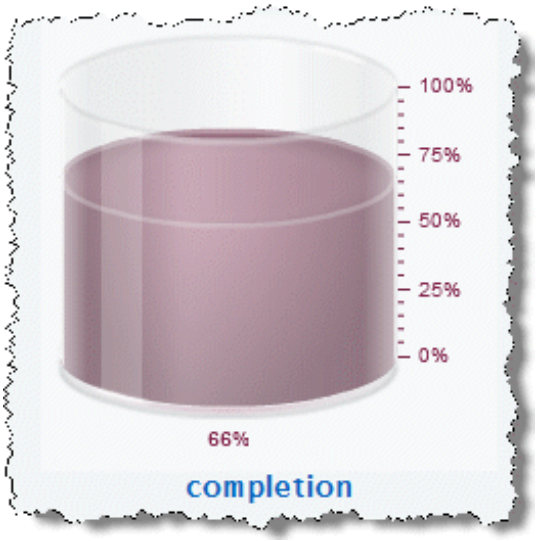
8. Preview the chart at any time.
9. Once you are satisfied, save the view:
    a. Click **Finish**.
    b. Give the view a name, and optional description.
    c. Click **Save**.

Cylinder Gauge

The cylinder gauge display a single numeric data value, or a set of numeric values, in relation to a scale rendered as a cylinder. The cylinder displays only one value at a time. With multiple values, users can page to a gauge for each value.

> **Note:** Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.

As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with a mix of gauge types.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.

Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with a cylinder gauge

1. Add a Gauge to This View.
2. Configure the Cylinder Gauge.
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view and once you are satisfied, save the view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

   Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.
2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

   Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.
3. Choose the type of gauge you want and complete configuration properties.

Configure the Cylinder Gauge

1. If needed, change the **Gauge Type** to `Cylinder`.
2. Define the scale for this gauge in the **Color Range**.

The scale defaults to 0-100, or a maximum close to the maximum value for this column. Unlike some other types of gauges, intermediate numbers in the scale do *not* affect the fill color for the cylinder.

Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest numbers that you want to show.

You can add numbers or remove numbers from the scale.

- To add a new number to the scale, enter a number and click **Add Value**.
- To remove a number, click on the number in the scale.

3. Enter the title for *this* specific gauge as the **Title**.
4. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.
5. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

   c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

   d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

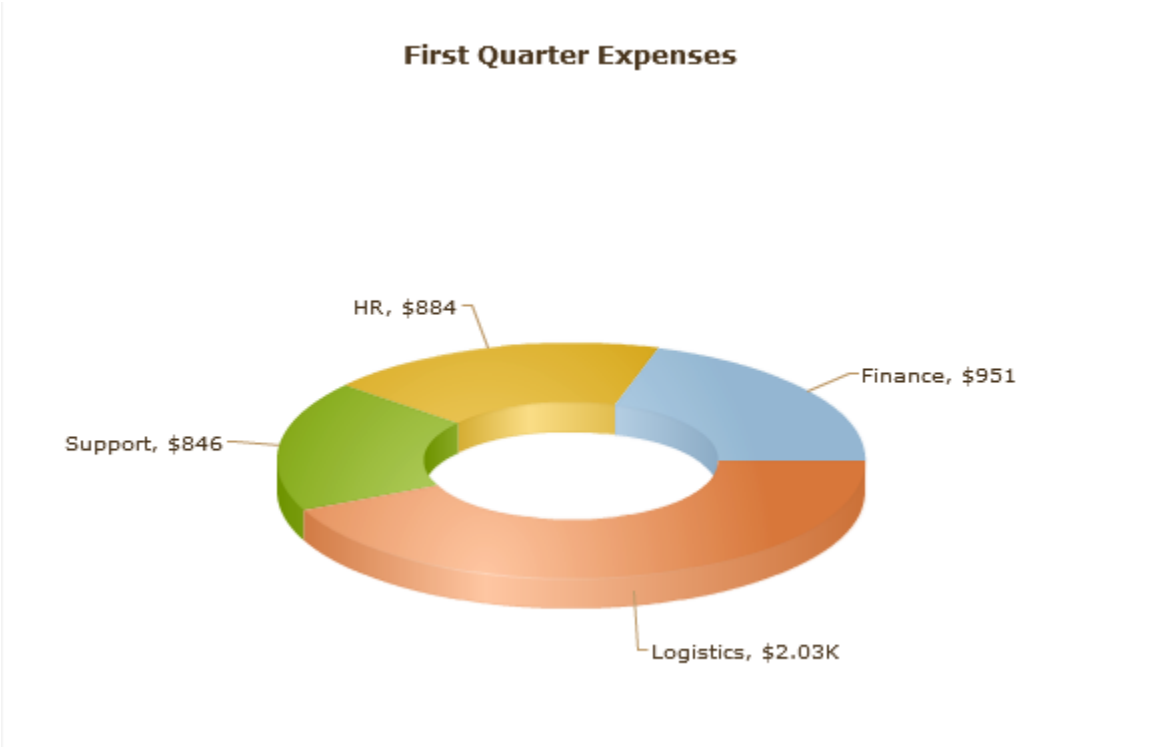
   e. Click **Add Color** to add this color to the list of colors used in the dial

   f. Add a color for each range of numbers you defined for the dial.

      If needed, simply click on a color in the list to delete that color.
6. Preview the gauge, as needed.

Doughnut Chart

Doughnut charts, like "Pie Chart" on page 986s, display a single numeric data point for multiple records as percentages of a whole. The lack of a center gives a different emphasis to the data.



There are two possible relationships between a slice and the records in the data set:

- *1-to-1* where each record represents one slice.
- *Aggregated* where an aggregate value of multiple records are shown as one slice.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what each slice is. In the example shown above, the Department field is the label. For 1-to-1 relationships, each record represents one slice and the value of the label should be unique in each record. With aggregate relationships, multiple records are aggregated into a slice and the value of the label may be repeated in different records. Each unique label value defines one slice. ■ One field with numeric data that defines the value (and size) of each slice. For 1-to-1 relationships, the value for each record is the value of the slice.

	With aggregate relationships, all the values of the records that match one unique label value are used to calculate the value of the slice based on the aggregate function you choose.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Doughnut Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the doughnut slices and drop this in the **Label** field.

   Tip: For better visual results, the total number of doughnut slices should be less than 75.

3. Drag the title of the column that contains numeric data into the **Value** field.

   Tip: For best results, the values for slices should be at least 3% or larger of the total.

4. Optionally, drag the title of the column with text values to use as the **Tooltip** when the mouse passes over each doughnut slice. If you omit this, tooltips simply do not appear in the chart.
5. Optionally, change the **Chart Visualization** property to set this as two dimensional (2D) or three dimensional (3D).
6. If the values of the **Label** field are not unique, change the **Aggregation** function from `None` to the function you want to use to calculate the value to use for each slice.
7. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

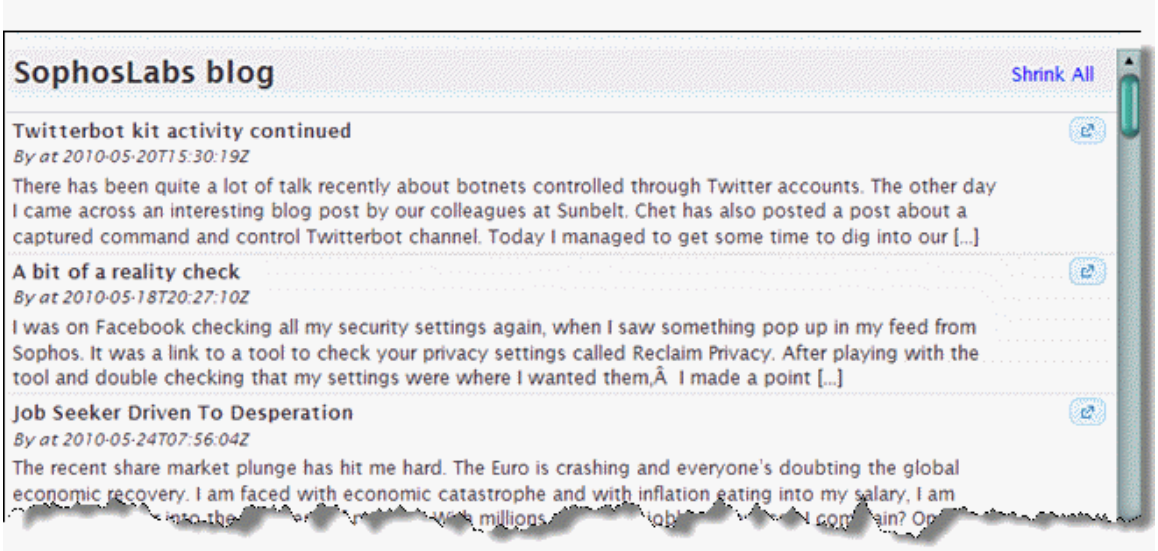
     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.

8. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
9. Preview the chart at any time.
10. Once you are satisfied, save the view:
    a. Click **Finish**.
    b. Give the view a name, and optional description.
    c. Click **Save**.

Feed Reader

This view displays web feeds, in RSS or Atom formats. It enables links back to the full content of each feed article or item.



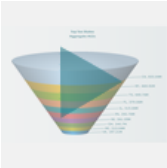
This view has a single configuration property, **Set to enable expand for record summaries**, which is set by default. When set, several lines of the article or description display for each item in the feed. User can choose to collapse this to a single line or expand it. If you clear this option, only a single line of the description displays in this view.

Characteristics

Data Requirements	One of the supported RSS or Atom formats.
Published Events	`link-click` = user clicks a link in cell. This applies specifically to those fields in RSS and Atom that are URLs.

Mobile Views	Yes

Funnel Chart



Funnel charts display a single series or data point for a set of records as a progression of slices of a three-dimensional funnel. Both the decreasing radius of the funnel and the overall depth of each slice indicate the decreasing size of the data.

See "Characteristics" on page 934 and "Configure This View" on page 935 for more information. For an alternate presentation of progressive numerical data, see "Pyramid Chart" on page 989.

There are two possible relationships between a slice and the records in the data set:

- *1-to-1* where each record represents one slice.
- *Aggregated* where an aggregate value of multiple records are shown as one slice.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies each slice. In the example shown above, the State field provides the label. For 1-to-1 relationships, each record represents one slice and the value of the label should be unique in each record. With aggregate relationships, multiple records are aggregated into a slice and the value of the label may be repeated in different records. Each unique label value defines one slice. ■ One field with numeric data that defines the value (and size) of each slice. Values shuld have some variation from large to small. In the example shown above, the aggregate AGI is the value. For 1-to-1 relationships, the value for each record is the value of the slice. With aggregate relationships, all the values of the records that match one unique label value are used to calculate

	the value of the slice based on the aggregate function you choose. For best results, the value for each record or aggregate value for each slice should be unique to ensure that slices do not misrepresent the data.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the Funnel view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for each slice and drop this in the **Label** field.

   Tip: For best results, limit the number of categories to 15 or less.

3. Drag the title of the column that contains numeric data into the **Value** field.
4. Optionally, drag and drop the title of the column with text values in the **Tooltip** field.
5. If the values of the **Label** field are not unique, change the **Aggregation** function from `None` to the function you want to use to calculate the value to use for each slice.
6. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
7. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
8. Preview the chart at any time.
9. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.

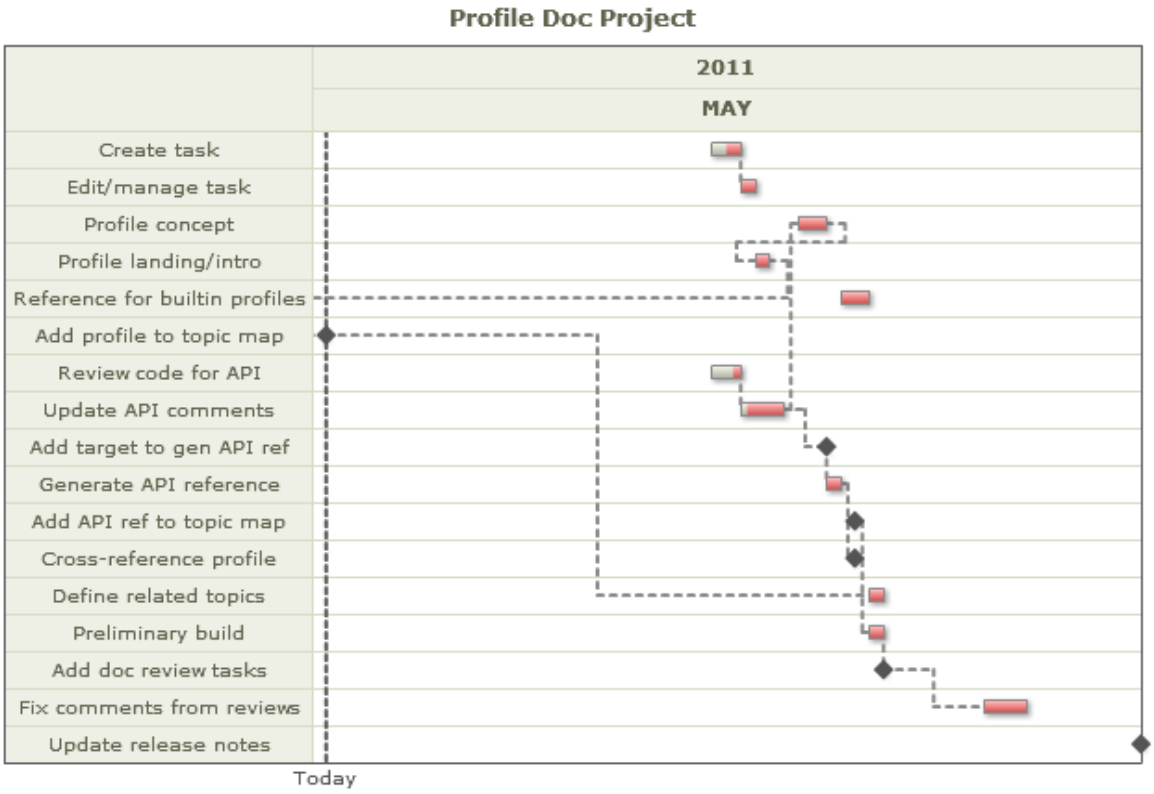
c. Click **Save**.

Gantt Chart

This view renders a Gantt chart tracking the progress of tasks across a timeline. It is typically used to render repeating 'items' with status information for projects.

This view automatically paginates large numbers of tasks, allowing users to see small sets of tasks and get additional pages to see more 'pages' of tasks. Tasks can also be grouped into *processes* if the data supports this.

See "Characteristics" on page 936 and "Configure This View" on page 937 for more information.



Characteristics

Data Requirements	A set of repeating items that represent the tasks to show in the chart. **Note:** For best results with medium to large numbers of tasks (more than 25), keep pagination in force in this view. If all tasks *must* remain within a single 'page,' limit the number of tasks.

	Tasks fields include: ■ A required text field with a title or summary of the task. ■ A required date field with the start date of the task. ■ A required date field with the end date of the task. ■ An optional numeric field with the percentage of completion for the task. Percentages must be a decimal number from 0 to 1 (such as 0.15) or integers with a percent sign (such as 15%). ■ An optional numeric field with a single task dependency. This number must correspond to the index number for the task that the current task depends on. ■ An optional text field for *processes* that group tasks with matching process values. ■ An optional text field for tooltips to include in task labels.
Published Events	■ `element-click` = published event when users click on a task rectangle in the timeline.
Mobile Views	No

Configure This View

1. If needed, change the path to the repeating items that represent tasks in **Select Dataset**.
2. Drag the title of a text column with the title or short description to the **Task** field.
3. Drag the title of a date column to the **Task Start Date** field.
4. Drag the title of a date column to the **Task End Date** field.
5. Complete any of the following optional fields:
   - **Predecessors**: drop the title of a numeric column with dependency information for tasks in this field. This must contain only a single integer that identifies the index number for a task that this task depends on.
   - **Percentage Completion**: drop the title of a numeric column with percentage of completion information for tasks in this field. Numbers must be either decimal, between 0 and 1, or in the form *nnn%* .
   - **Processes**: drop the title of an alphanumeric column with grouping categories for tasks in this field. All tasks with the same category appear in a single row of the chart.

- **Tasks Tooltip**: drop the title of an alphanumeric column with alternate text for task tooltips in this field.

The visual effects of these fields provide different combinations. For more information, see "Visual Effects of Optional Data Configuration for Gantt Charts" on page 938.

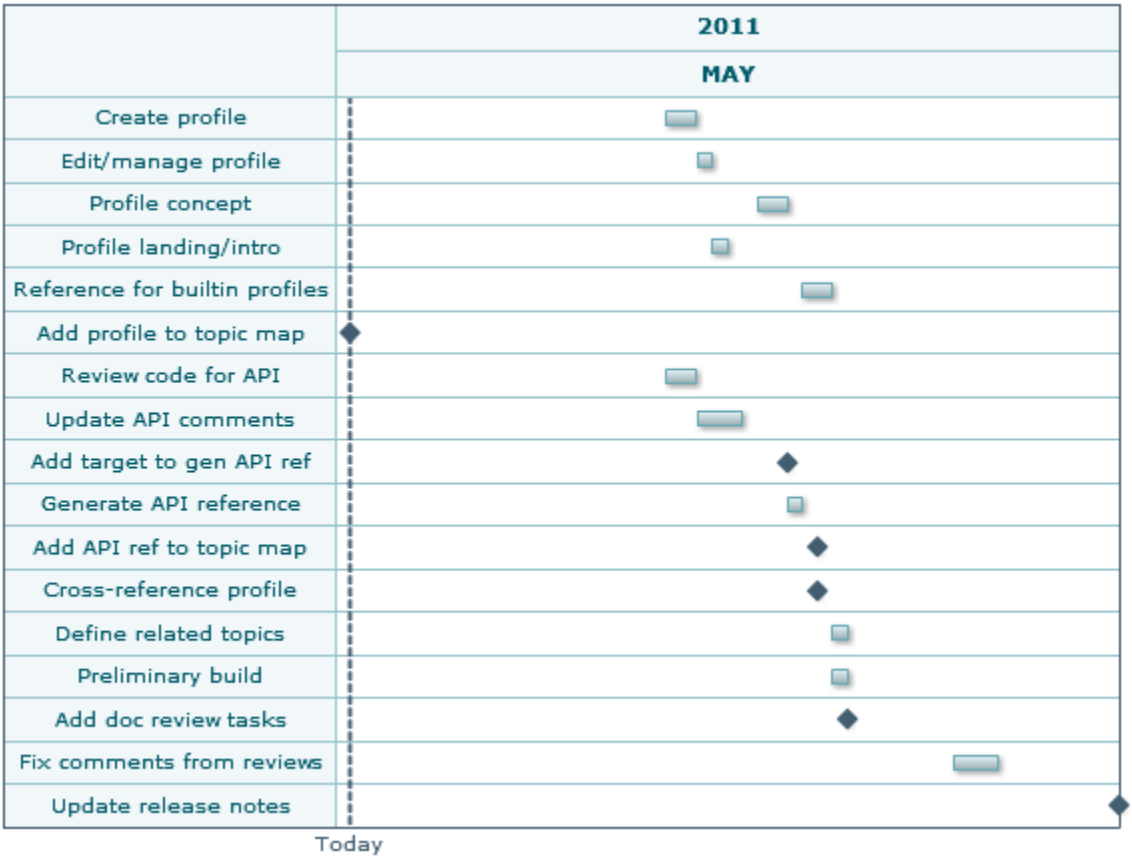
6. If needed, change the format to match the date format used in task start and end dates. This default to `mm/dd/yyyy`.
7. Click **Next** and set an optional **Caption** or **Subcaption** for the chart.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
8. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
9. Preview and save the view.

Visual Effects of Optional Data Configuration for Gantt Charts

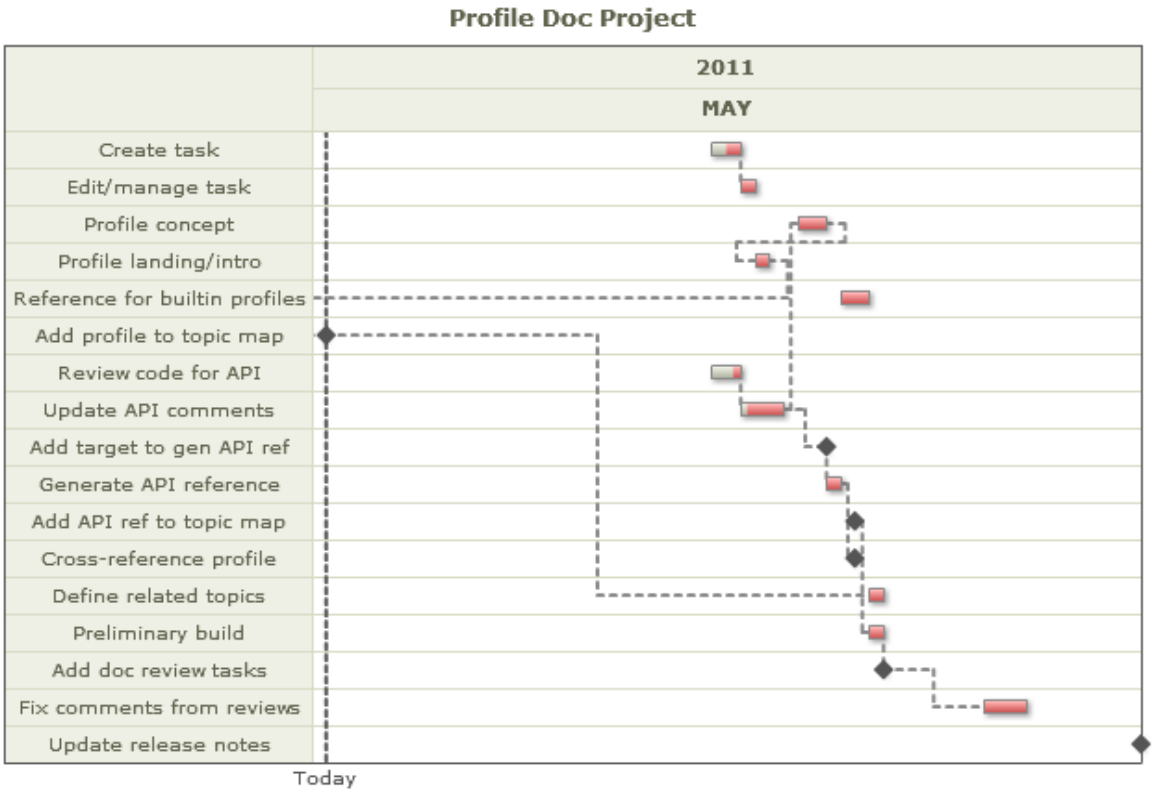
The most minimal Gantt chart provides task descriptions and start/end dates, as shown in this example:

Minimal Gantt Chart with Task, Start and End Date Only



The next example shows the effect of adding pecentage of completion and dependency data to the chart:

Minimal Gantt Chart Plus % of Completion and Dependencies

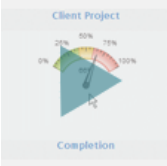


The **Processes** field allows you to group tasks, but also affects labelling and tooltips for tasks. The **Tasks Tooltip** field affects tooltips and behaves differently when combined with Processes. The effects of these two fields are:

- **Using Just Processes**
  - Tasks are grouped under each Process, so the left column displays the Process value rather than the task description.
  - Each task bar in the timeline has a label showing both the task description and the start/end date range.
  - The tooltip for each task matches the task bar label.
- **Using Just Tooltip**
  - Tasks are on individual lines, so the left column displays the Process value rather than the task description.
  - Each task bar in the timeline has a label showing just the tooltip value.
  - The tooltip for each task shows both the tooltip value and the start/end date for the task.
- **Using Both Processes and Tooltips**

- Tasks are grouped under each Process, so the left column displays the Process value rather than the task description.
- Each task bar in the timeline has a label showing the task description.
- The tooltip for each task shows just the tooltip value.

Gauges



Gauges display a single numeric data point in relation to a scale. They are used in many cases to illustrate the status of some metric.

Note: Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.

The scale for gauge views can be rendered in several different formats:

- Dial Gauge
- Bulb Gauge
- Bullet Gauge also sometimes called a bullet graph
- Cylinder Gauge
- LED Gauge
- Linear Gauge
- Thermometer Gauge

You can choose to include several gauges in one view, with different gauge types for each.

Geo Map

Use this view to display one set of numeric statistics for one or more countries within a map of a specific region or of the entire world. Numeric values display as a gradient of a color, from light to dark corresponding to lowest to highest values.

See "Characteristics" on page 942 and "Configure This View" on page 942 for more information. To map multiple statistics for countries, see the "Intensity Map" on page 966 view.

Characteristics

Data Requirements	■ A set of repeating items that each must contain: ■ A field with either: ■ ISO 3166-1 county codes (alpha-2) ■ ISO 3166-1 country names ■ ISO 3166-2 regional subdivision names These codes and names are maintained by the International Standards Organization for continents, regions and countries of the world. See the "ISO 3166-1 Alpha 2 Country Codes" on page 943 section for more information and suggestions. ■ At least one other field with a number to use as the intensity statistic to render in the map.
Published Events	■ `select` = a user clicks on one of the countries displaying an intensity in the map ■ `region-click` = a user clicks on one of the regions in the map
Mobile Views	No

Configure This View

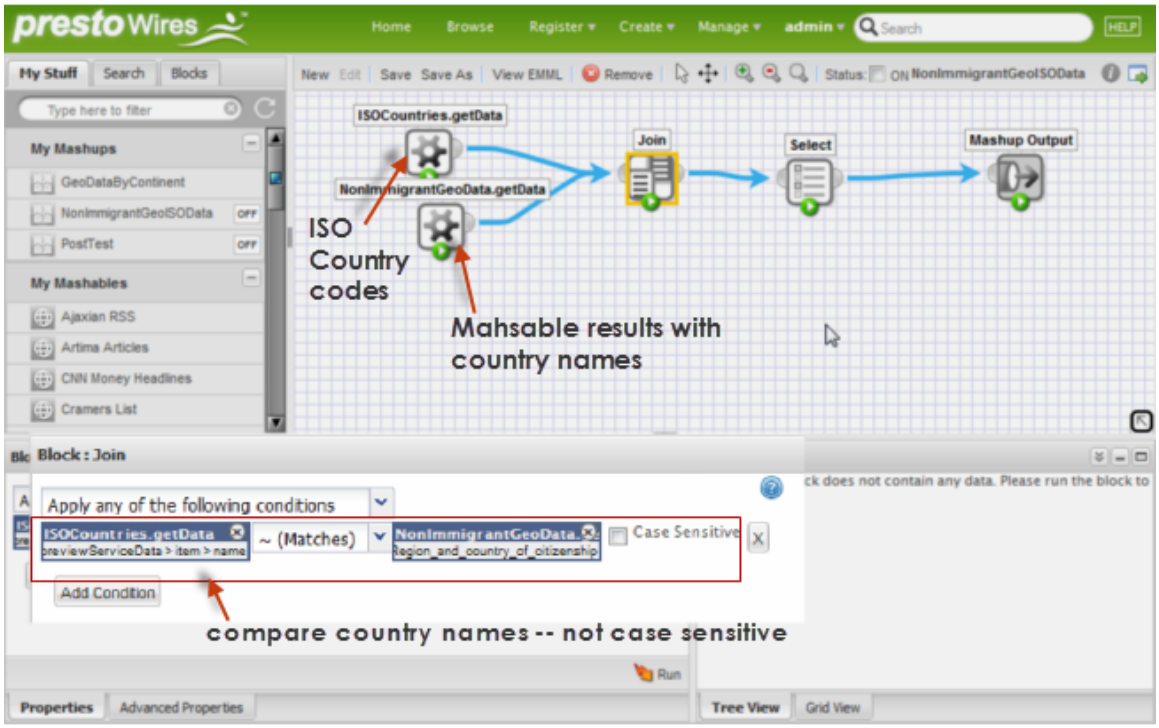
1. If needed, change the path to the items you want to map in **Select Dataset** in **Configure Data**.
2. Drag the field that contains either ISO 3166-1 county codes (alpha-2), ISO 3166-1 country names or ISO 3166-2 regional subdivisions to the **Country Column**. See "ISO 3166-1 Alpha 2 Country Codes" on page 943 for suggestions to ensure you have this data.
3. Drag one field with statistics to render in the map to the **Column** field.

4. Choose the area of the map to use in the **Region** field. This defaults to `World`, but you can select a specific continent, specific regions within a continent or a specific country.
5. Preview the map and save the view.

ISO 3166-1 Alpha 2 Country Codes

The most common problem you may encounter using this view is that the results for your mashable or mashup have country, region or continent names that do not match the ISO 3166-1 standards. One solution is to add ISO 3166-1 Alpha-2 country codes or valid country names to your results and use this field in the the view. You can find these country codes and names in English at http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm.

The easiest solution is to download these codes as an XML file (from the previous link) and register this XML file as a mashable information source. You can then use this mashable in mashups to add the specific ISO country codes to the results of other mashables or mashups, such as this example:



This mashup uses the Join block to compare the country codes from the results of a mashable to the country names in the ISO county code data. ISO country names are fully capitalized, so it is usually better not to do case sensitive comparisons.

Google Map

Use this view to show one or more repeating items in your data as locations in a map with an overlay of roads and streets.

This view supports the default *map type* for Google with a flat map showing roads. It also supports the satellite map type and a hybrid map type. Google maps support up to 20 levels of zoom and configurable markers for each location.

Note: The Earth map type is no longer supported for Google Map views.

See "Characteristics" on page 944 and "Configure This View" on page 944 for more information on the requirements to use this view.

Characteristics

Data Requirements	■ Internet access ■ A set of repeating items that each must contain either latitude and longitude data as separate fields or a single field with valid location data such as a street, city, state/region and country.
Published Events	`marker-click` = a user clicks on one of the location markers in the map
Mobile Views	Yes

Configure This View

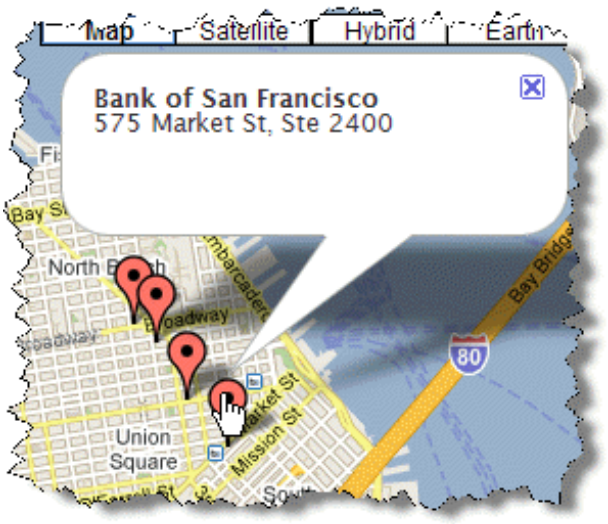
To configure this view

1. Configure the coordinates for items in **Configure Data**.
   a. If needed, change the path to the items you want to map in **Select Dataset**.
   b. Set the field(s) that provide coordinates for each item:

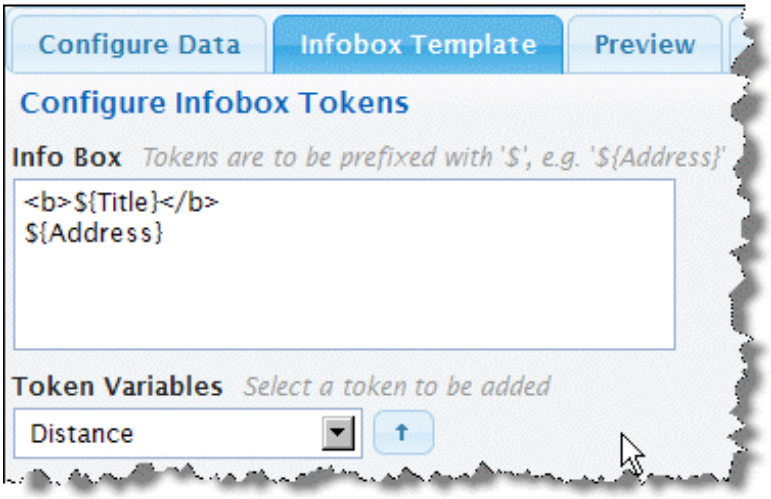
      Choose **Latitude** and set both the latitude and the longitude.

      Or choose **Point** and select the field that contains a full valid location.

2. If you want to have tooltips with additional information for the mapped item markers, complete the **Infobox Template** step. For example:



Build a template for the content that should appear in the marker. Template combine *tokens*, in the form `${path/to/field/data}`, that identify fields from each item to supply data for this item, literal characters and optionally HTML markup. Templates can contain multiple lines, as needed. For example:



a. If needed, delete the default token.

b. Enter literal characters wherever you want within the overall template. If you need multiple lines, simply start a new line where needed.

c. To add data from fields in the item, select the path to the field you want and click ↑.

d. Add HTML markup as raw tags, such as `<b>This will be bold</b>`.

e. Continue adding literal characters, HTML markup and tokens until you are satisfied with the template.

3. Preview and save the view.

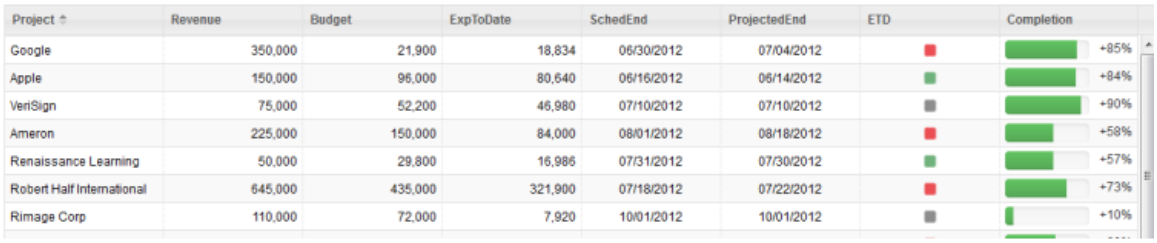
Grid and KPIs View

The grid view displays repeating data in a table of any size for desktop and mobile devices. It provides a wide variety of formatting options for data from simple font changes to inline spark charts or KPI rules to display status indicators. It automatically paginates data.

Note: In previous releases, this view did not wrap content into additional lines when the data was longer than the column width. In this release, the Grid View now wraps cell content.

You can use view configuration to:

- Add Columns and Assign Values or Move or Remove Columns.
- Change Column Headings, Width and Text Formatting.
- Apply many different formats to columns based on the type of data. See "Show As: Change the Column Format for Different Types Data" on page 949 .
- Use Templates to Combine Data, Add Literal Text or Add HTML Tags or .
- Add Key Performance Indicators (KPI) to apply formatting based on conditions you define.



Project	Revenue	Budget	ExpToDate	SchedEnd	ProjectedEnd	ETD	Completion
Google	350,000	21,900	18,834	06/30/2012	07/04/2012		+85%
Apple	150,000	96,000	80,640	06/16/2012	06/14/2012		+84%
VeriSign	75,000	52,200	46,980	07/10/2012	07/10/2012		+90%
Ameron	225,000	150,000	84,000	08/01/2012	08/18/2012		+58%
Renaissance Learning	50,000	29,800	16,986	07/31/2012	07/30/2012		+57%
Robert Half International	645,000	435,000	321,900	07/18/2012	07/22/2012		+73%
Rimage Corp	110,000	72,000	7,920	10/01/2012	10/01/2012		+10%

Characteristics

Data Requirements	A set of repeating items with one or more fields. For columns with trend information displayed as a Spark Chart, see "Data for Spark Chart Columns" on page 954 for additional requirements.
Published Events	■ `rowclick` = user clicks in a cell to select a data row in the table

	■ `cellclick` = reserved for future use ■ `headerclick` = user clicks in a cell in the column header row
Mobile Views	Yes

Configure This View

MashZone NextGen defines a default *dataset*, the path to the repeating items in your data to use for each row of the table, in the **Data Configuration** step. You can choose a different dataset, if needed.

Then define which columns to use in this table, the data that should appear in each column and the formats to apply to each column. You can:

- Move or Remove Columns
- Add Columns and Assign Values
- Change Column Headings, Width and Text Formatting
- Show As: Change the Column Format for Different Types Data, such as formatting numbers, links or using spark charts or bullet graphs.
- Use Templates to Combine Data, Add Literal Text or Add HTML Tags
- Add Key Performance Indicators (KPI)

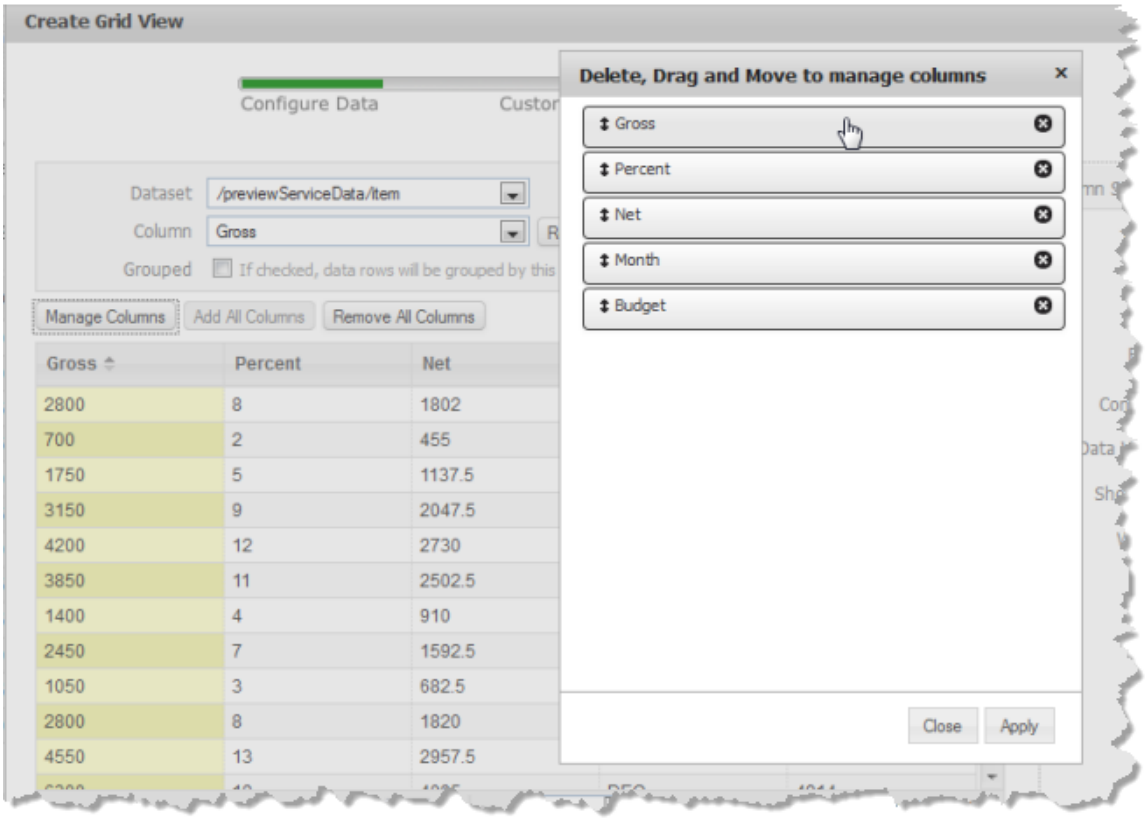
Preview the results and once you are satisfied, save the view.

See also "Precedence for Column Settings and KPI Rules" on page 966.

Move or Remove Columns

- To remove columns from the table, either:
  1. Select that column and click **Remove Column**.
  2. Click **Manage Columns** and click for each column you want to remove.
  3. Click **Remove All Columns** and add back only those columns you want. See "Add Columns and Assign Values" on page 948 for instructions.

  Removing a column hides the data in the view and also removes this data from the `rowclick` event . You can, however, still combine this data with other data for other columns using a template. Or use this data in KPI Rules.
- To move a column, click **Manage Columns**. Select a column, drag it in the list of columns and drop it where you want it.



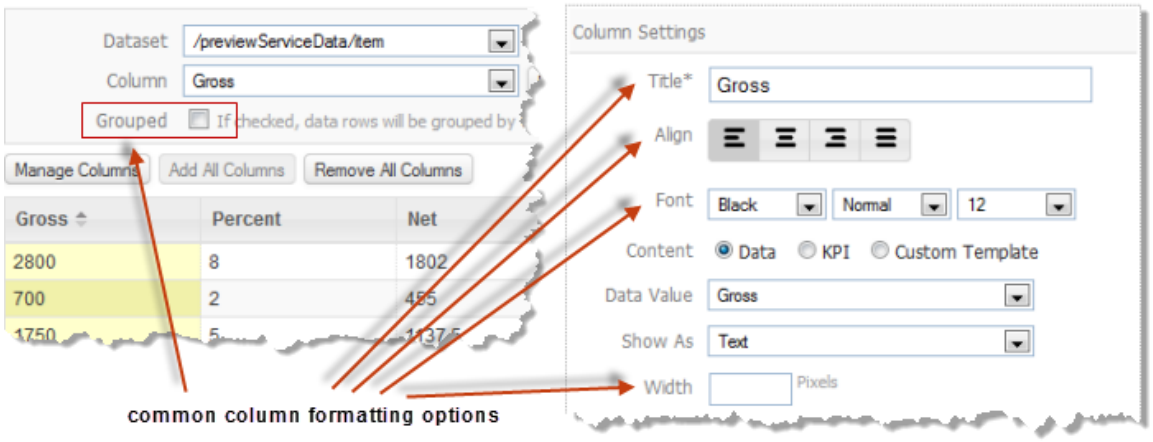
Add Columns and Assign Values

To add a column to the grid

1. Click **Add Column**.
2. Then assign data using **Data Value**, Use Templates to Combine Data, Add Literal Text or Add HTML Tags or Add Key Performance Indicators (KPI).
3. Optionally, enter a heading in the **Title** property.
4. Set any other optional formatting for this column. See "Change Column Headings, Width and Text Formatting" on page 948 or "Show As: Change the Column Format for Different Types Data" on page 949 for information.

Change Column Headings, Width and Text Formatting

Select a column from the preview area or use the **Column** list and use any of these common formatting options:



- Use **Title** to change the column heading.
- Use **Align** to change the alignment of data in this column.
- Use **Font** options to change the color, variation and size of the data for this column.
- Drag column widths in the preview area or use **Width** to directly control the number of pixels to use for this column.
- Use **Grouped** to group rows with the same value for this column together. Each group has a visual subtitle with the value for that group.

Note: Both alignment and font formatting can be overridden if you also use a custom template or KPI Rules for Cells or Rows. See "Precedence for Column Settings and KPI Rules" on page 966 for more information.

Show As: Change the Column Format for Different Types Data

Select a column in the preview area or use the **Column** list. Use **Show As** to apply other formats based on the type of data in this column:

- `Date` = expects a string with a date, time or date and time. It converts this string to a date and optionally a time for sorting purposes.

  You choose the format to display the date in the **Date Format** property.

  Note: In previous releases, you could also conditionally apply color formatting or arrow icons to dates and numbers. With this release, Use KPI Cell Formatting instead.

  Color and arrow formatting in views created in previous releases is supported. To edit this, update the view to use KPI Rules.

- `HTML` = for data that contains HTML tags.
- `Image` = shows images in this column when the data is URLs to image files.

You can also determine the **Height** or **Width** to use for all images in this column. Both default to `auto` which uses the actual size of each image. To set a size, enter the number of pixels for each dimension.

- `Link` = creates a link for this column when the data is URLs.

  By default, `Click here` displays as the text of the link. Set the **Show link text** option to change this to:

  - A single phrase of your own choosing. Simply enter the phrase in the field below

    If this field has a list of columns, click to return to edit mode where you can enter a value.

  - The dynamic value of a column that you select for each row. To set dynamic values, click to get different values from another column. Then select the column to use from the list of available columns.

- `Number` = allows you to apply common numeric formats if the data is numeric. See "Number Formatter" on page 951 for details.

  > **Note:** In previous releases, you could also conditionally apply color formatting or arrow icons to dates and numbers. With this release, Use KPI Cell Formatting instead.
  >
  > Color and arrow formatting in views created in previous releases is supported. To edit this, update the view to use KPI Rules.

- `Text` = displays the data as is. This is the default.

  You can also limit the number of characters to show in this column and append an ellipsis (...). Set the **Show only** option and enter the maximum number of characters.

- `Text (Show...)` = limits the text to 80 characters and appends an ellipsis (...) to indicate that there is more content. You can also get limited character content using `Text` and the **Show only** option

- `SparkLine` = displays a sparkline chart to show a trend if the data for this column is multiple, numeric values. See "Spark Charts, Bullet Graphs and Percentages" on page 951 for instructions.

- `SparkColumn` = displays a spark column chart to show a trend if the data for this column is multiple, numeric values. See "Spark Charts, Bullet Graphs and Percentages" on page 951 for instructions.

- `Spark Win/Loss` = displays a spark win/loss column chart to show a trend if the data for this column is multiple, numeric values. See "Spark Charts, Bullet Graphs and Percentages" on page 951 for instructions.

- `Discrete` = displays a spark discrete column chart to show a trend if the data for this column is multiple, numeric values. See "Spark Charts, Bullet Graphs and Percentages" on page 951 for instructions.

- `Bullet` = displays a bullet graph chart to show progress compared to a target for columns with single numeric values. See "Spark Charts, Bullet Graphs and Percentages" on page 951 for instructions.
- `Percentage` = displays a percentage bar for columns with single numeric values. See "Spark Charts, Bullet Graphs and Percentages" on page 951 for instructions.

Number Formatter

For columns with numeric data, set the **Number Formatter** option to define these common numeric properties:

Number of decimal places	The number of decimal places to display for every number in this column. If you do not set this property, numbers with a decimal value are shown as is.
Prefix numbers with this text	A currency symbol or any other literal text characters to prefix each number in this column.
Suffix numbers with this text	A currency abbreviation or any other literal text characters to use as a suffix for each number in this column.
Show negative values with ( )	Surround negative values with parentheses symbols. This is typically used in accounting formats for credits.
Show negative values in red	Show negative values in a red color.
Show thousands comma separator	Add commas as thousand separators to numbers greater than 999.

See also "Use KPI Cell Formatting" on page 963 for ways to conditionally apply colors or add icons, such as arrows, to numeric data.

Spark Charts, Bullet Graphs and Percentages

You can display several different types of *mini-charts* in each cell of columns for numeric data to make this data easier to understand:

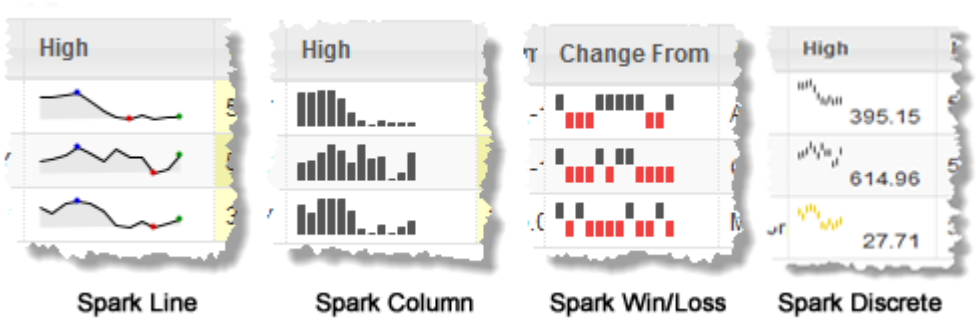
- Spark Charts: Line, Column, Win/Loss or Discrete: these charts graphically show a trend in a highly compressed format. They require that the data for this column contain multiple numeric values for each row of the dataset.

- Bullet Graphs: display a numeric value in relationship to a numeric target that you define.
- Percentage Bars: display a numeric value between 0 and 100 as a percentage of a horizontal bar.

Spark Charts: Line, Column, Win/Loss or Discrete

Spark charts graphically show a distilled view of a trend for a column that contains multiple numeric values for each row. Typically, this is a trend over some time period.

You can choose from these Spark Chart formats:



Since Spark Charts require multiple values for a single column, the data for that column must be either comma-separated values or XML. See "Data for Spark Chart Columns" on page 954 for details.

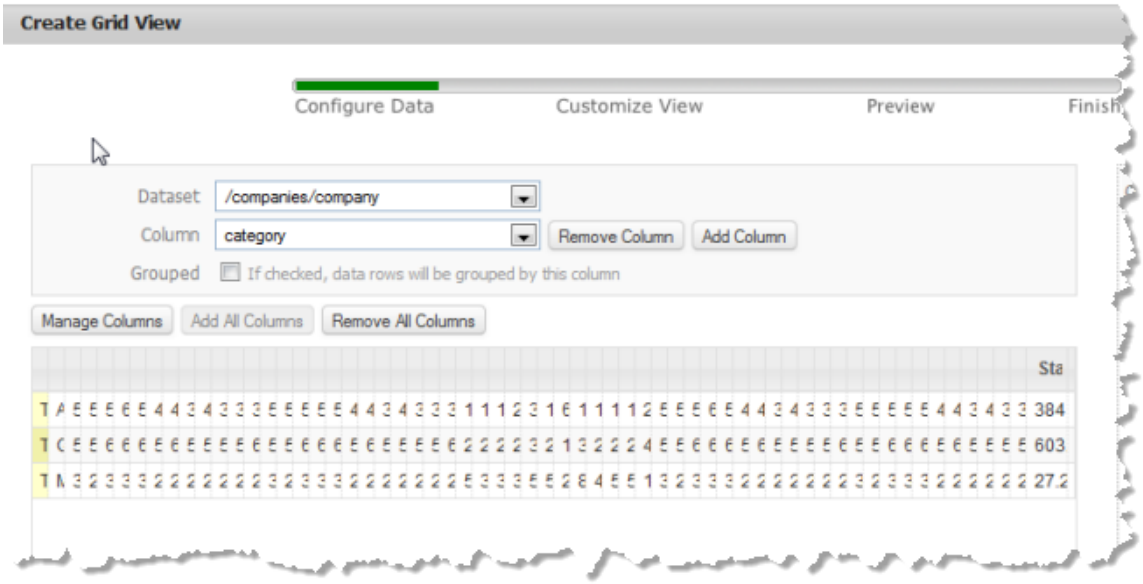
To display Spark Charts for *comma-separated values*

1. Select a column with numeric CSV data in the preview area or the **Column** list.
2. Change **Show As** to the Spark Chart format you want to use.
3. If needed, Change the Chart Size.
4. Set the **Show Latest Value** option if you wish to show the last value in this column as well as the Spark Chart.
5. For Discrete Spark Charts, optionally set a **Threshold Value** to display the chart lines in an alternate color if any values are below the threshold. Either:
   - Enter the single value to use as the threshold for all charts in this column.

     If this field has a list of columns, click to return to edit mode where you can enter a value.
   - Click to get different threshold values from another column. Then select the column to use as the threshold from the list of available columns.

To display Spark Charts for *XML structures*

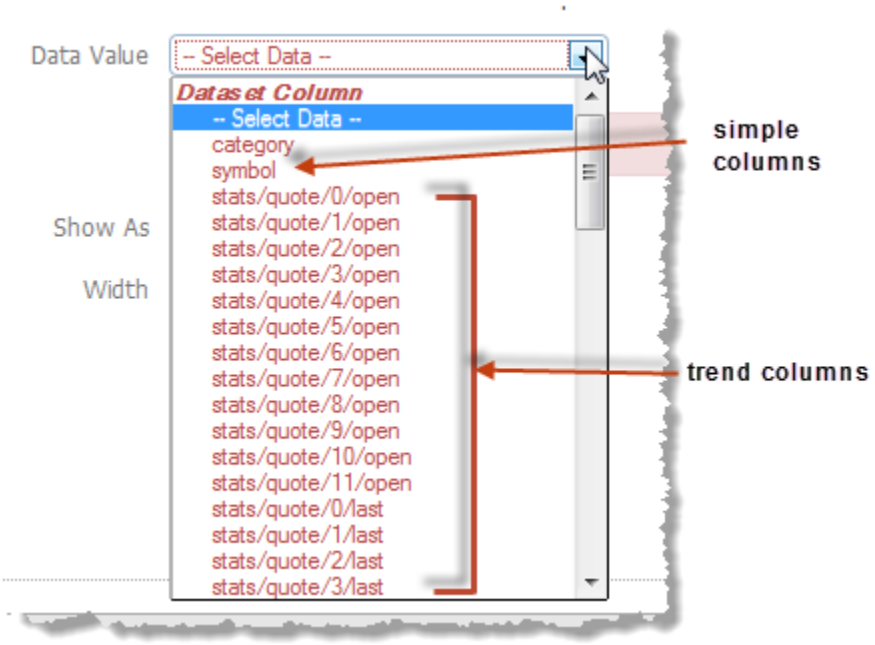
1. In many cases, the initial grid preview for XML structures with multiple values in a column is not useful, such as this example:



When this happens, it is better to remove all the default columns and add back only those columns you want:

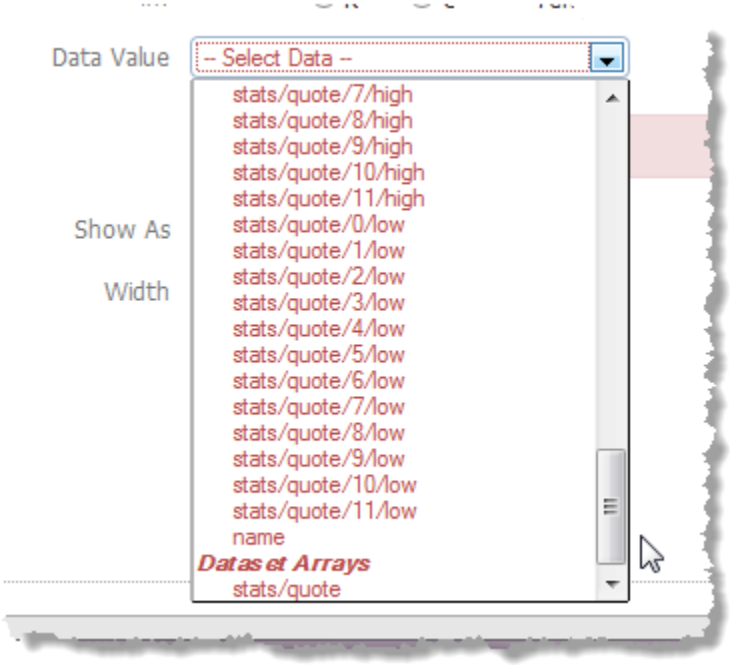
a. Click **Remove All Columns**.

b. Add columns for any simple data that you want, skipping any of the columns listed that contain trend data for the chart. See "Add Columns and Assign Values" on page 948 for instructions.



2. Click **Add Column** to add a column for trend data and the Spark Chart you want to use.

3. In **Data Value**, scroll to the bottom of the list of columns to the `Dataset Arrays` section and select the repeating item with trend data. For this example, that is `stats/quote`:



4. In **Series Column**, choose the trend column you want to display as a Spark Chart.
5. Choose the type of Spark Chart you want to use in **Show As**.
6. If needed, Change the Chart Size.
7. Set the **Show latest value** option if you want to display both the last value for this trend column along with the Spark Chart.
8. For Discrete Spark Charts, optionally set a **Threshold Value** to display the chart lines in an alternate color if any values are below the threshold. Either:
   - Enter the single value to use as the threshold for all charts in this column.

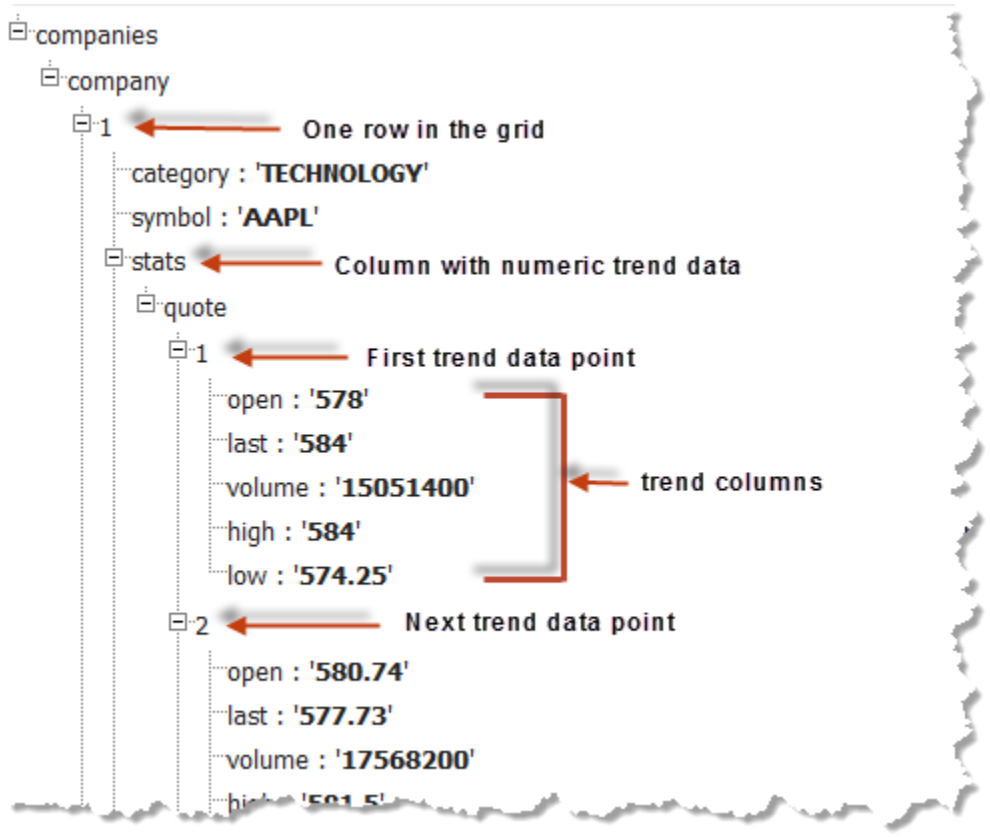
     If this field has a list of columns, click to return to edit mode where you can enter a value.
   - Click to get different threshold values from another column. Then select the column to use as the threshold from the list of available columns.

Data for Spark Chart Columns

To show Spark Charts in a column, the column must contain multiple values in one of these formats:

- Comma-separated values, such as this:

- An XML structure, such as this:



In this case, each row of the grid is populated from a set of repeating items, such as `company` in this example. Each item, in turn, has a child that wraps trend data, such as the `stats` field. This wrapper is the parent to another set of repeating items, `quote` in this example, with trend data you can use for Spark Charts. Each repeating `quote` has one or more fields (the trend columns) with the numeric data for one point in the trend, such as the `open` field in this example.

The actual XML for this example looks something like this:

Bullet Graphs

Bullet graphs graphically show progress for a column with one numeric value per cell as compared to a target number. For example:

You can enter a single target value or compare the values for this column to values in another column, such as comparing actual to a budget.

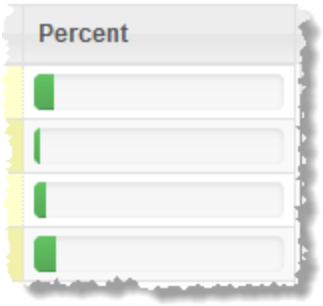
To use bullet graphs for a column in a grid

1. Select a column with numeric values in the preview area or the **Column** list.
2. Change **Show As** to `Bullet`.
3. If needed, Change the Chart Size.
4. If desired, set the **Show Value** option.
5. Optionally, change the **Performance Color** (the fill color of the bar showing current progress for each bullet graph).
6. Enter or choose the **Target** for the graph:
   - Enter the single value to use one target for all bullet graphs in this column.

     If this field has a list of columns, click to return to edit mode where you can enter a value.
   - Click to get have target values comes from another column. Then select the column to use as the target from the list of available columns.
7. Optionally, change the percentages for the **Lower Range**, **Middle Range** or **Upper Range** to change the background gradient for the graph.

Percentage Bars

For percentage bars, the column must contain a single numeric value between 0 and 100. For example:



You can also choose to show the value in addition to the percentage bar.

1. Select a column with numeric values in the preview area or the **Column** list.
2. Change **Show As** to `Percentage`.
3. If needed, set the **Show Value** option.

Change the Chart Size

For Spark Charts or Bullet Graphs, you can change the size (both width and height) of the chart that is displayed in each row. By default **Chart Size** is set to `Small`. Simply select another size.

Note: If you set the chart size, you can override the default width of the chart with the **Width** property. See "Change Column Headings, Width and Text Formatting" on page 948 for more information.

A chart size of `Auto` fits the height of the chart into the existing height of each row. All the other chart sizes may adjust the height of the row to fit the chart height. For example:

Chart Size = auto

Chart Size = medium

Use Templates to Combine Data, Add Literal Text or Add HTML Tags

You can combine data from several fields into a single value, add literal text or add HTML tags to data and assign the result to a column using *templates*. Templates combine tokens, identifying a field, with optional literal characters or HTML markup to build the data and format it.

Note: HTML within the template can override other formatting options such as font size or alignment.

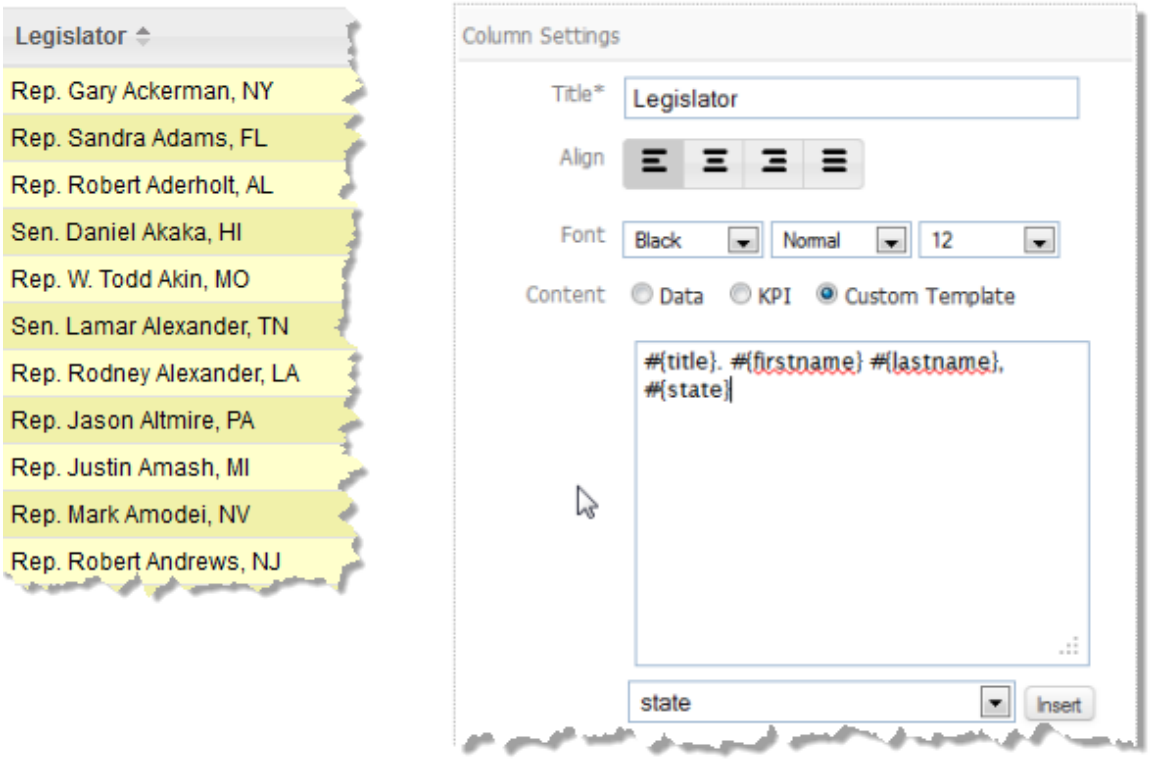
Tokens to identify a field are shown in the form `#{`*path/to/the/field*`}`.

Select the column from the list where you want to combine data and:

1. Set the **Custom Template** option for the content.

2. If needed, delete the default token which defines the field currently assigned to this column.
3. Enter literal characters or HTML markup wherever you want within the overall template.

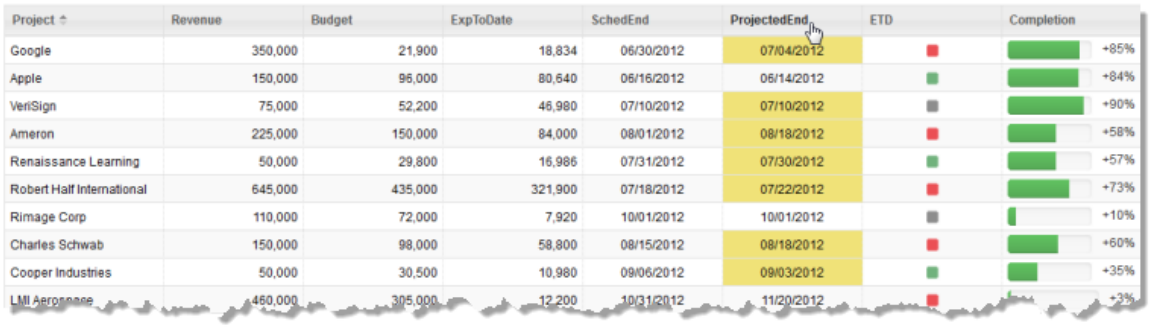
   In this example, both the period and the comma in the template are literal characters and `#{title}` is a token for the legislator's title field.



4. To add data from other fields to the template, select the path to the field you want and click **Insert**.
5. Continue adding literal characters, HTML markup and tokens for other fields until you are satisfied with the template.

Add Key Performance Indicators (KPI)

Key performance indicators, or KPIs, are formats that can be conditionally applied to individual cells in a column or individual rows in the grid to indicate status or performance. You define the rules that determine whether formatting is applied and choose the formatting to apply if data meets the requirements of a given rule.



Project	Revenue	Budget	ExpToDate	SchedEnd	ProjectedEnd	ETD	Completion
Google	350,000	21,900	18,834	06/30/2012	07/04/2012		+85%
Apple	150,000	96,000	80,640	06/16/2012	06/14/2012		+84%
VeriSign	75,000	52,200	46,980	07/10/2012	07/10/2012		+90%
Ameron	225,000	150,000	84,000	08/01/2012	08/18/2012		+58%
Renaissance Learning	50,000	29,800	16,986	07/31/2012	07/30/2012		+57%
Robert Half International	645,000	435,000	321,900	07/18/2012	07/22/2012		+73%
Rimage Corp	110,000	72,000	7,920	10/01/2012	10/01/2012		+10%
Charles Schwab	150,000	98,000	58,800	08/15/2012	08/18/2012		+60%
Cooper Industries	50,000	30,500	10,980	09/06/2012	09/03/2012		+35%
LMI Aerospace	460,000	305,000	12,200	10/31/2012	11/20/2012		+3%

The previous example show two types of KPI formatting: icons in one column to indicate the expected delivery status for projects (on-time, late or early) and background colors in specific cells in another column indicating projects expected to finish in the current quarter.

To use KPI formatting for a grid

1. Select a column in the preview area or the **Column** list where you want to apply KPI formatting. For row formatting. select the column that should be used in rules to apply the KPI formatting.
2. Set the **KPI** option for the content.

3. Click **Add KPI Rule** in the bottom right corner of Data Configuration.

   The View Maker wizard opens a KPI Rules area below the grid preview and Column Settings with a blank rule for this column.

   a. Complete the blank condition for this rule. See "Define a KPI Condition" on page 961 for instructions.

   b. If needed, add other conditions and set condition matching appropriately. See "Set Multiple KPI Conditions" on page 961 for instructions.

   c. Choose the KPI formatting to apply when the condition(s) are met. See "Use KPI Icons" on page 961, "Use KPI Cell Formatting" on page 963 or "Use KPI Row Formatting" on page 964 for instructions.

4. Click **Apply** to save this rule and preview the effect.

See "Manage Rules and Conditions" on page 965 for other common KPI tasks.

Define a KPI Condition

KPI conditions compare the values of one column to a single value or to the data in another column. You define the column to compare, the comparison to make and the value(s) to compare to in the:

- *Left Field* = the column with data to compare. Typically this is the column you have already selected in the preview area, but you can select another column if needed.
- *Middle Field* = comparison to make.
  - `is equal to`, `is not equal to`, `is less than`, `is less than or equal to`, `is greater than` and `is greater than or equal to` can be used for numeric or date data.

    Note: For correct numeric comparisons, both columns (left and right) in the condition *must* be numbers. In many cases, columns contain numbers but are considered `Text`. The easiest way to ensure this is to set **Show As** to `Number`.

    If both columns are not numeric, the comparison treats both as text which can cause errors.
  - `contains` works for text fields and will match as long as the characters are found anywhere in column values. Comparison is *case sensitive*.
- *Right Field* = a single literal value or the name of another column with the values to compare to. Either:
  - Enter the single value to use in this field.

    If this field has a list of columns, click to return to edit mode where you can enter a value.
  - Click to get different values from another column. Then select the column to use from the list of available columns.

Set Multiple KPI Conditions

Click **Add** next to an existing condition to add another condition. Complete the fields (see "Define a KPI Condition" on page 961 for more information) for this new condition.

Change the condition matching rule to determine what is considered a match. Choose either `Match All Conditions` or `Match Any Conditions`.

Use KPI Icons

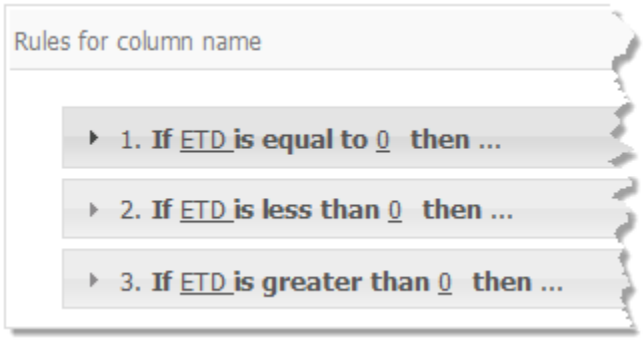
KPI Icons are useful to quickly define status or progress for a column based on conditions, such as the ETD column in this example:

Project	Revenue	Budget	ExpToDate	SchedEnd	ProjectedE	ETD	Completion
Google	350,000	21,900	18,834	06/30/2012	07/04/2012		+85%
Apple	150,000	96,000	80,640	06/16/2012	06/14/2012		+84%
VeriSign	75,000	52,200	46,980	07/10/2012	07/10/2012		+90%
Ameron	225,000	150,000	84,000	08/01/2012	08/18/2012		+58%
Renaissance Learning	50,000	29,800	16,986	07/31/2012	07/30/2012		+57%
Robert Half International	645,000	435,000	321,900	07/18/2012	07/22/2012		+73%
Rimage Corp	110,000	72,000	7,920	10/01/2012	10/01/2012		+10%
Charles Schwab	150,000	98,000	58,800	08/15/2012	08/18/2012		+60%
Cooper Industries	50,000	30,500	10,980	09/06/2012	09/03/2012		+35%
LMI Aerospace	460,000	305,000	12,200	10/31/2012	11/20/2012		+3%

Page 0 of 2 5 View 1 - 20 of 25

The ETD column indicates whether the expected delivery date for a project is on-time , early or late .

This example uses three rules, one for each icon:



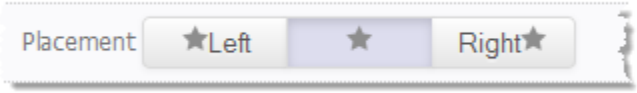
You can have just the icons show or show both the data and the icons. You can also repeat icons to indicate rating or importance or use two or more different icons to indicate different conditions.

To use icons for KPI formatting

1. Set the **Icon** option.
2. Click **Select Icon** and either:
   - Select one of the built-in KPI icons and click **Close**.

- Or click the **Custom URL** tab to use your own icon. Enter the URL to the icon image and click Close.

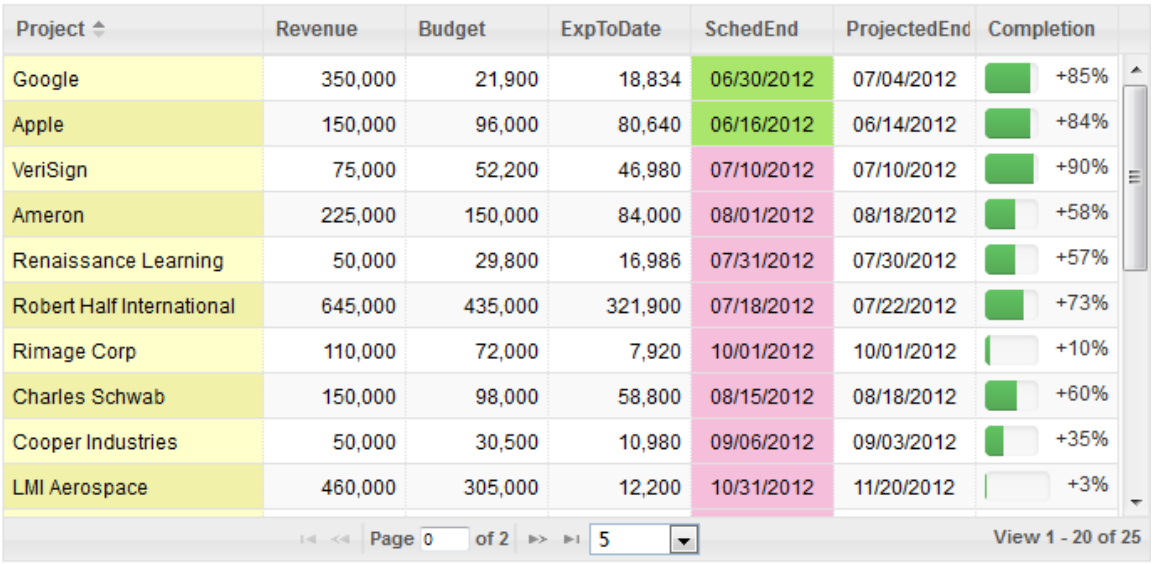
3. If you want to repeat this icon more than once, such as a rows of stars to indicate a rating, change the **Repeat** value.
4. Choose the icon placement and data visibility. Left or Right includes both the icon and the data. Centered shows just the icon.



5. To show other icons for this condition, click **Add** and repeat the steps to choose the icon, repetition and placement for another icon.
6. Click **Apply**.

Use KPI Cell Formatting

KPI cell formatting can change the background color, font color, font variation or font size for data in the selected column, such as this example:



Project	Revenue	Budget	ExpToDate	SchedEnd	ProjectedEnd	Completion
Google	350,000	21,900	18,834	06/30/2012	07/04/2012	+85%
Apple	150,000	96,000	80,640	06/16/2012	06/14/2012	+84%
VeriSign	75,000	52,200	46,980	07/10/2012	07/10/2012	+90%
Ameron	225,000	150,000	84,000	08/01/2012	08/18/2012	+58%
Renaissance Learning	50,000	29,800	16,986	07/31/2012	07/30/2012	+57%
Robert Half International	645,000	435,000	321,900	07/18/2012	07/22/2012	+73%
Rimage Corp	110,000	72,000	7,920	10/01/2012	10/01/2012	+10%
Charles Schwab	150,000	98,000	58,800	08/15/2012	08/18/2012	+60%
Cooper Industries	50,000	30,500	10,980	09/06/2012	09/03/2012	+35%
LMI Aerospace	460,000	305,000	12,200	10/31/2012	11/20/2012	+3%

Page 0 of 2 5 View 1 - 20 of 25

This uses different background colors to indicate projects scheduled to be finished in the first two quarters versus projects scheduled for the last two quarters.

Note: KPI cell formatting overrides both font formatting set in Column Settings and font formatting set in KPI rules with row formatting .

To use KPI cell formatting

1. Set the **Highlight Cell** option.
2. Set any of the font options, as needed.
3. If needed, click **Background** and find the color you want to use as the background. Click **Set this Color**.
4. Click **Apply**.

Note: Cell formatting overrides formatting from Column Settings. In turn, Cell formatting can be overridden by other KPI Rules with Row formatting. See "Precedence for Column Settings and KPI Rules" on page 966.

Use KPI Row Formatting

KPI row formatting can change the background color, font color, font variation or font size for data in rows that match the condition(s) in the rule, such as this example which highlights project rows when the expected revenue is greater than $250,000:

Project	Revenue	Budget	ExpToDate	SchedEnd	ProjectedEnd	Completion
Google	350,000	21,900	18,834	06/30/2012	07/04/2012	+85%
Apple	150,000	96,000	80,640	06/16/2012	06/14/2012	+84%
VeriSign	75,000	52,200	46,980	07/10/2012	07/10/2012	+90%
Ameron	225,000	150,000	84,000	08/01/2012	08/18/2012	+58%
Renaissance Learning	50,000	29,800	16,986	07/31/2012	07/30/2012	+57%
Robert Half International	645,000	435,000	321,900	07/18/2012	07/22/2012	+73%
Rimage Corp	110,000	72,000	7,920	10/01/2012	10/01/2012	+10%
Charles Schwab	150,000	98,000	58,800	08/15/2012	08/18/2012	+60%
Cooper Industries	50,000	30,500	10,980	09/06/2012	09/03/2012	+35%
LMI Aerospace	460,000	305,000	12,200	10/31/2012	11/20/2012	+3%

Page 0 of 2 5 View 1 - 20 of 25

Note: KPI row formatting overrides font formatting set in Column Settings. It can be overridden, however by font formatting set in KPI rules with cell formatting .

To use KPI cell formatting

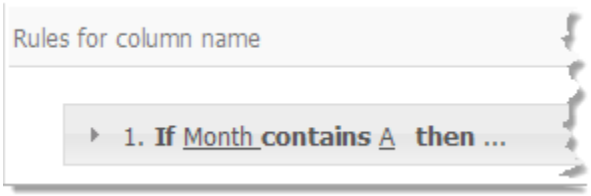
1. Set the **Highlight Row** option.
2. Set any of the font options, as needed.
3. If needed, click **Background** and find the color you want to use as the background. Click **Set this Color**.
4. Click **Apply**.

Note: Row formatting overrides Cell formatting from other KPI Rules or column settings. See "Precedence for Column Settings and KPI Rules" on page 966.

Manage Rules and Conditions

You can also:

- Open or collapse rules by clicking on the rule heading.



- Remove a rule. Click the rule heading to open it, if needed, and click **Remove**.

- Remove a condition. Open the rule, if needed, and click **Remove** next to the condition.

> **Note:** You cannot remove the very first condition. Instead, simply change the fields to match another rule and then delete the duplicate.

Precedence for Column Settings and KPI Rules

You can set font properties in Column Settings as well as in KPI Rules for columns or for rows. KPI Rules for columns and rows both also allow you to set the background color respectively.

When multiple formats are defined for the same visual property, the format with the highest precendence is applied:

1. KPI Rules with Row formatting overrides
2. KPI Rules with Cell formatting, which overrides
3. Column Settings

Intensity Map

Use this view to display one or several numeric statistics for one or more countries within a map of a specific region or of the entire world. Numeric values display as a gradient of a color, from light to dark corresponding to lowest to highest values. Each set of statistics display in a separate tab.

See "Characteristics" on page 966and "Configure This View" on page 967 for more information. See also the "Geo Map" on page 941 view which supports a single statistic with a similar rendering.

Characteristics

Data Requirements	■ A set of repeating items that each must contain: ■ A field with either: ■ ISO 3166-1 county codes (alpha-2) ■ ISO 3166-1 country names ■ ISO 3166-2 regional subdivision names

	These codes and names are maintained by the International Standards Organization for continents, regions and countries of the world. See "ISO 3166-1 Alpha 2 Country Codes" on page 967 for more information and suggestions. ■ At least one other field with a number to use as the intensity statistic to render in the map.
Published Events	`select` = a user clicks on one of the countries displaying an intensity in the map
Mobile Views	No

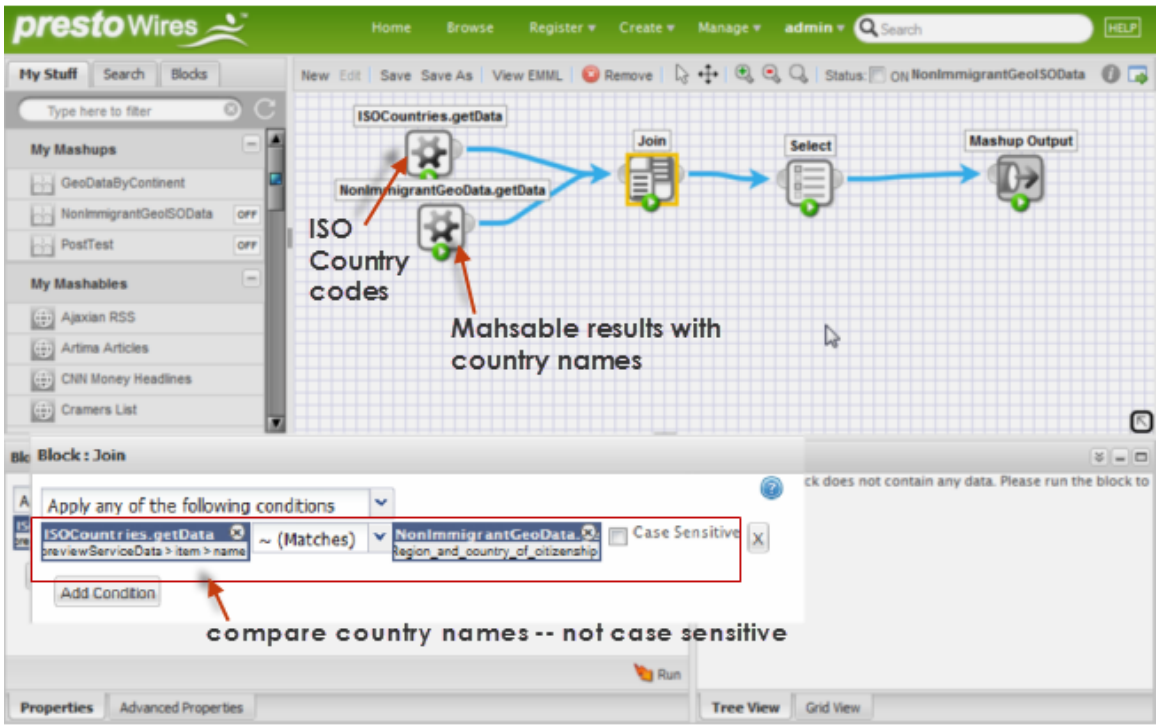
Configure This View

1. If needed, change the path to the items you want to map in **Select Dataset** in **Configure Data**.
2. Drag the field that contains either the ISO 3166-1 county codes (alpha-2), ISO 3166-1 country names or ISO 3166-2 regional subdivisions to the **Country Column**. See "ISO 3166-1 Alpha 2 Country Codes" on page 967 for suggestions on how to ensure you have this data.
3. Drag one or more fields with statistics to render in the map to the **Columns** field. Each field you add will create a separate tab with its own map.
4. Choose the area of the map to use in the **Region** field. This defaults to `World`, but you can select a specific continent, specific regions within a continent or a specific country.
5. Preview the chart at any time.
6. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

ISO 3166-1 Alpha 2 Country Codes

The most common problem you may encounter using this view is that the results for your mashable or mashup have country, region or continent names that do not match the ISO 3166-1 standards. One solution is to add ISO 3166-1 Alpha-2 country codes or valid country names to your results and use this field in the the view. You can find these country codes and names in English at http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm.

The easiest solution is to download these codes as an XML file (from the previous link) and register this XML file as a mashable information source. You can then use this mashable in mashups to add the specific ISO country codes to the results of other mashables or mashups, such as this example:

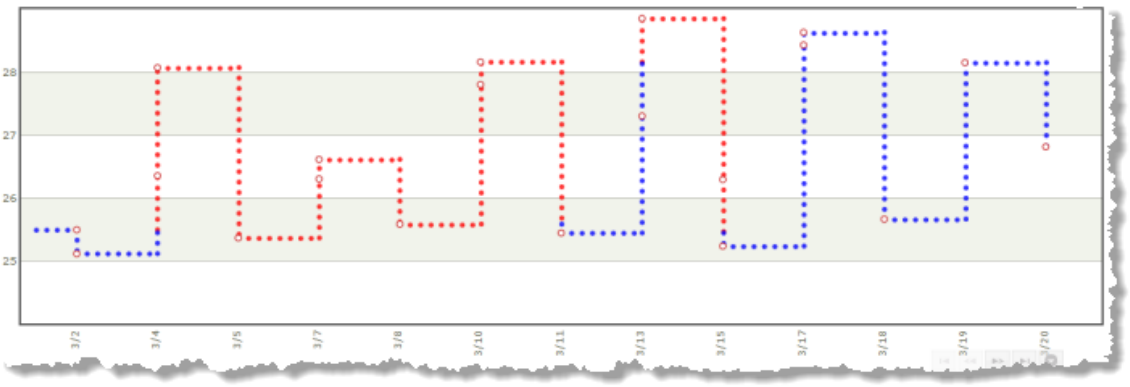


This mashup uses the Join block to compare the country codes from the results of a mashable to the country names in the ISO county code data. ISO country names are fully capitalized, so it is usually better not to do case sensitive comparisons.

Kagi Chart

Kagi Charts display a stepped line indicating the change in trend (up or down) for one numeric dimension. Color indicates a change in trend that exceeded thresholds. The thresholds for this chart default to the last high value or low value, but can also be set as a percentage of the range or values or as absolute values.

A common example for a Kagi chart is a line for price that reflects changes in supply and demand. Or changes in stock, commodity or currency prices.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what is on each tick of the X axis. ■ One field with numeric data that is plotted as a stepped line.
Mobile Views	Yes

Configure This View

Choose the Kagi Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column with labels for the dimension you want to plot and drop it in the **Label** field.
3. Drag the title of the numeric column for the dimention you want to plot into the **Value** field.
4. Optionally, drag the title of a text column into the **Tooltip** field.
5. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.

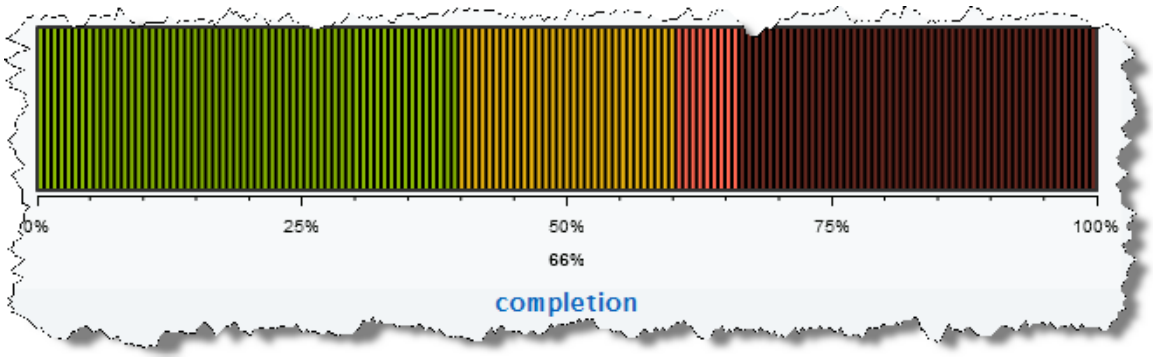
6. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
7. Preview the chart at any time.
8. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

LED Gauge

LED gauges display a single numeric data point in relation to a scale that is rendered as LED lights. This scale can also progress through color ranges to provide additional status information. The scale can be either horizontal or vertical. Only one value is displayed at a time. With multiple values, users can page to this gauge for each value.

Note: Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.

As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with a mix of gauge types.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.

Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with an LED gauge

1. "Add a Gauge to This View" on page 971.
2. "Configure the LED Gauge" on page 971
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view and once you are satisfied, save the view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

   Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.
2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

   Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.
3. Choose the type of gauge you want and complete configuration properties.

Configure the LED Gauge

1. If needed, select the Series column for the gauge and change the **Gauge Type** to `LED`.
2. Define the scale and color range for this gauge in the **Color Range**.

The scale defaults to 0-100, or a maximum close to the maximum value for this column and a single color for the entire range.

Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest numbers that you want to show.

You define the number of colors that appear in the scale by how many intermediate numbers you include in the scale. With no intermediate numbers, the scale is one color. Add an intermediate number to change the scale color at that point.

You can add numbers or remove numbers from the scale.

- To add a new number to the scale, enter a number and click **Add Value**.
- To remove a number, click on the number in the scale.

3. Use the **Chart Orientation** property to choose the direction for this scale.
4. Enter the title for *this* specific gauge as the **Title**.
5. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.
6. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

   c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

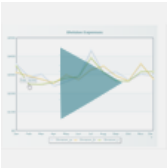
   d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

   e. Click **Add Color** to add this color to the list of colors used in the dial

   f. Add a color for each range of numbers you defined for the dial.

      If needed, simply click on a color in the list to delete that color.
7. Preview the gauge, as needed.

Line Chart



Basic line charts display several numeric data points (series or columns) for a set of records. Each data point for a given record is connected by a line. This illustrates the change between data points and the lines emphasize the differences between records for the same data point.

Note: You can also create Real-Time Line Charts for event mashables that dynamically update as events occur.

Line charts are also related to Scatter Charts and Area Charts. For a reverse relationship between data points and records, see Sparkline. For datasets with a large number of records, see "Zoom Line Chart" on page 1023.

For more information on using the Line Chart view, see "Characteristics" on page 973 and "Configure This View" on page 973.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what is on each tick of the X axis. This is the *category* field. In the example shown above, the Month field is the category. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines a line in one color. In the example shown above, there are three series for Division_a, Division_b and Division_c.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Line Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a line in a different color to the chart.

   Tip: For best results, limit the number of series (the number of lines per category) to 12 or less.

   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, "Change the Series Plot to Column, Line or Area" on page 905 or "Change Series Labels" on page 904 to make the chart easier to understand.

4. Optionally, change the **Chart Scroll** to yes to allow the categories to spill over to additional pages when you have larger numbers of categories.
5. If desired, Add a Target Line to the chart.
6. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
7. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
8. Preview the chart at any time.
9. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Real-Time Line Chart

Real-time line charts display numeric measures for Event mashables as lines that illustrate the change in events over time. This chart also updates automatically as events occur. You add real-time charts to workspaces in Mashboard. For more information, see "What are real-time charts?" on page 74.

Note: Before you can use a real-time chart, MashZone NextGen administrators must create event mashables and connections to Apama and/or the Event Bus.

See "Characteristics" on page 975 and "Configure a Real-Time Line Chart" on page 975 for more information on using the Real-Time Line Chart.

Characteristics

Data Requirements	An event source with events from the Event Bus or from an Apama scenario. Events must contain at least: ■ One text or date field to plot on the X axis. ■ One or more measures with numeric data to be plotted in a line.
Published Events	None
Mobile Views	Yes

Configure a Real-Time Line Chart

The configuration wizard opens with the first step to Select a Data Source. Then, you can:

- Define Date Columns
- Assign Columns to Axes
- Set Column Properties
- Set Chart Properties

Select a Data Source

To find and select the event mashable or Apama mashable that you want to use as the data source for this chart, either:

- Select the event source for the mashable you want from the list and click **Next**.

  Or

- Enter part of the name for the event source to filter the list. Select the event source you want and click **Next**.

Select a Data Source

To find and select the event mashable or Apama mashable that you want to use as the data source for this chart, either:

- Select the event source for the mashable you want from the list and click **Next**.

  Or

- Enter part of the name for the event source to filter the list. Select the event source you want and click **Next**.

Define Date Columns

If none of the columns for this event or Apama mashable are dates or times, click **Next**.

Note: Date configuration information is not required for the `Event_LastUpdateTime_Internal` column included in Apama mashable events.

If there are columns that are dates:

1. Set the **Date** option for that column.

   If the data for this column is numeric, MashZone NextGen treats this as the number of milliseconds since January 1, 1970 to determine a date and time.

2. If the data for a date column is text, you must also:

   a. Enter a pattern as the **Format** for this data to allow MashZone NextGen to correctly interpret the date. Use the following characters to define the pattern:

Character	Meaning
y or Y	Year If the data contains a four digit year, the format pattern can contain any number of `y` characters to recognize the year component of the date. If the data contains only a one or two digit year, a format pattern of `yy` returns a four digit year while a format pattern of `yyyy` returns just the year.
M	Month If the date has a text abbreviation of the month, use `MMM` as the format pattern. If the month is numeric, use `MM` or `M` as the format pattern.
d or D	Day
h or H	Hour
m	Minute
s or S	Second or Milliseconds

Character	Meaning
a or A	AM/PM
- or /	As the date component separator
:	As the time component separator
space or other literals	Can be used within the format pattern and appear exactly as typed.

b. Select the **Language** for this data.

3. Once you have identified all columns that are dates or times, click **Next**.

Assign Columns to Axes

You assign event columns to the axes of the chart and optionally partition events into subcategories to determine how event data will be plotted in the chart. To assign a column, select one of the columns from the list on the left and drop it into a field. To delete a column drag the column out of a field and drop it.

With measure columns that you assign to the Y axis, you must also choose an aggregate function to apply to the measure: `Sum`, `Min`, `Max`, `Avg` or `Count`.

There are three patterns you can follow to assign columns:

- **Dynamic Time Line for One or More Measure Across All Events** aggregates and plots all events along a growing time line, as events occur. Each measure you choose to include generates one line.

  To create a dynamic time line across all events:

  - Assign a date column to the **X-axis**.

    For Apama mashables, this is the `Event_LastUpdateTime_Internal` column.

  - Assign one or more numeric columns to the **Y-axis**. Choose an aggregate function to apply to each column to determine how event data is combined.

  - Skip the **Partition** field.

- **Dynamic Time Line for One Measure Partitioned by Category** plots all events on a growing time line. Events are partitioned by a category, each unique category is plotted as a separate line for one measure.

  To create a dynamic, partitioned time line chart:

  - Assign a date column to the **X-axis**.

    For Apama mashables, this is the `Event_LastUpdateTime_Internal` column.

- Assign a single numeric column to the **Y-axis** and choose the aggregate function to apply.
- Assign a text column to the **Partition** field. This column defines the partitions.

- **Dynamic Categories** plots event data across categories on the X axis. There is no timeline, but as events occur, the lines on the chart are completely replotted.

  For example, you can plot temperature, humidity and other weather measures for different cities using the Apama Weather sample. The category is city.

  To create a dynamic categories chart:

  - Assign a text column to the **X-axis**. The unique values of this column represent one point on the line.
  - Assign one or more numeric columns as measures to the **Y-axis**. Each column generates a separate line.

    Choose an aggregate function to apply to each measure.
  - Skip **Partition**.

Set Column Properties

You can change labels and visual formats for any column:

- Set a **New Name** to change the label that appears in tooltips and the chart legend, if any.
- Select a **Format** pattern for numeric or enter a pattern for date columns. You can:
  - Choose a pattern to set the decimal and thousand place separators as well as the total number of decimal places for numeric columns.

    For example, choose:

    ```
    1,234.56
    ```

    To see numbers with two decimal places, a period as the decimal point and a comma as the thousand separator.
  - Set the format and separators to use with dates and times.

    Use these symbols:

Character	Meaning
y or Y	Year If the data contains a four digit year, the format pattern can contain any number of `y` characters to recognize the year component of the date.

Character	Meaning
	If the data contains only a one or two digit year, a format pattern of `yy` returns a four digit year while a format pattern of `yyyy` returns just the year.
M	Month If the date has a text abbreviation of the month, use `MMM` as the format pattern. If the month is numeric, use `MM` or `M` as the format pattern.
d or D	Day
h or H	Hour
m	Minute
s or S	Second or Milliseconds
a or A	AM/PM
- or /	As the date component separator
:	As the time component separator
space or other literals	Can be used within the format pattern and appear exactly as typed.

- Turn on rounding for numeric columns to round values to zero decimals.

Click **Next**.

Set Chart Properties

Finally, set general properties for the chart including:

- Enter a **Chart title** if desired.
- Change the default value of the **X-axis name** to display a label for this axis. This defaults to the label for the column assigned to this axis.

  You can also delete the value to remove this label from the chart.

- Change the default value of the **Y-axis name** to display a label for this axis. If you assigned a single measure to this axis, this defaults to the label for that measure. If there are multiple measures for this axis, the label defaults to `Value`.

  You can also delete the value to remove this label from the chart.
- Choose the **Line type**:
  - `Linear`
  - `Curved`
  - `Step-after`
- Set **Legend** to display a legend for the chart.

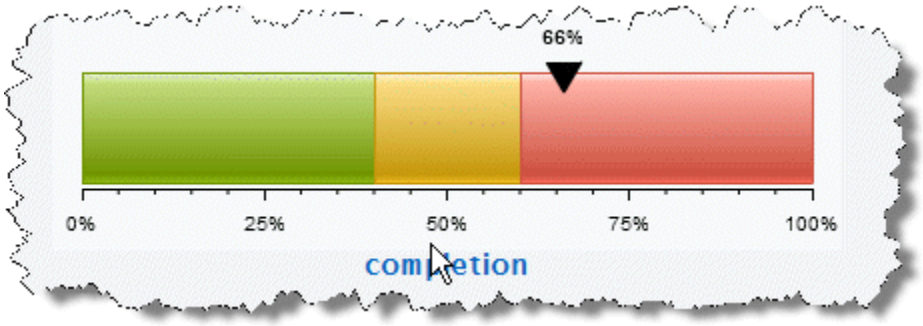
Click **Save**. The chart begins to plot data and updates periodically as events occur.

Linear Gauge

Linear gauges display a single numeric data point, or a set of numeric data points, in relation to a scale that renders as a horizontal bar. The bar can also progress through color changes to provide additional status information.

> **Note:** Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.

As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with a mix of gauge types.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.

Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with a linear gauge

1. Add a Gauge to This View.
2. Configure the Linear Gauge
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view and once you are satisfied, save the view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

   Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.
2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

   Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.
3. Choose the type of gauge you want and complete configuration properties.

Configure the Linear Gauge

1. If needed, select the Series column for the gauge and change the **Gauge Type** to `Linear`.
2. Define the scale and color range for this gauge in the **Color Rangel**.

The scale defaults to 0-100, or a maximum close to th maximum value for this color with a single color.

Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest numbers that you want to show.

You define the number of colors that appear in the scale by how many intermediate numbers you include in the scale. With no intermediate numbers, the scale is one color. Add an intermediate number to change the scale color at that point.

You can add numbers or remove numbers from the scale.

- To add a new number to the scale, enter a number and click **Add Value**.
- To remove a number, click on the number in the scale.

3. Enter the title for *this* specific gauge as the **Title**.
4. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.
5. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

   c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

   d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

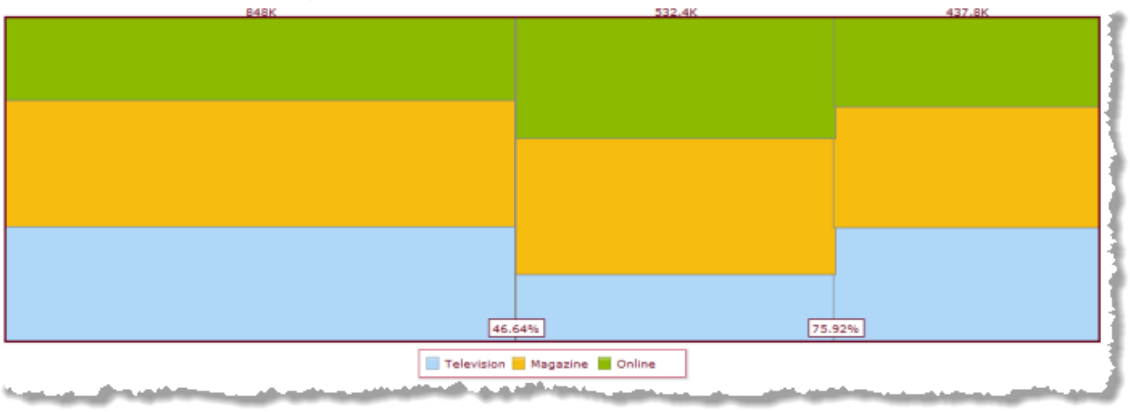
   e. Click **Add Color** to add this color to the list of colors used in the dial

   f. Add a color for each range of numbers you defined for the dial.

      If needed, simply click on a color in the list to delete that color.
6. Preview the gauge, as needed.

Marimekko Chart

The Marimekko view displays multiple numeric dimensions as stacked columns allowing easy comparison across different categories. The heights for each column is shown as a percentage of the entire height so that the overall column height does not vary.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies the name for each column. This is the *category* field. ■ Two or more fields with numeric data that you add as a *series*. Each series defines the bars that are graphed with one color.
Mobile Views	Yes

Configure This View

Choose the Marimekko Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

   Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least two columns that contains numeric data into the **Series** field. Each column you add to Series adds another stack of bars in a different color to each column in the chart.

   Tip: For best results, limit the number of series (the number of bars or columns per interval/category) to eight or less for best results.

It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

If needed, "Change Series Labels" on page 904 to make the chart eaiser to understand.

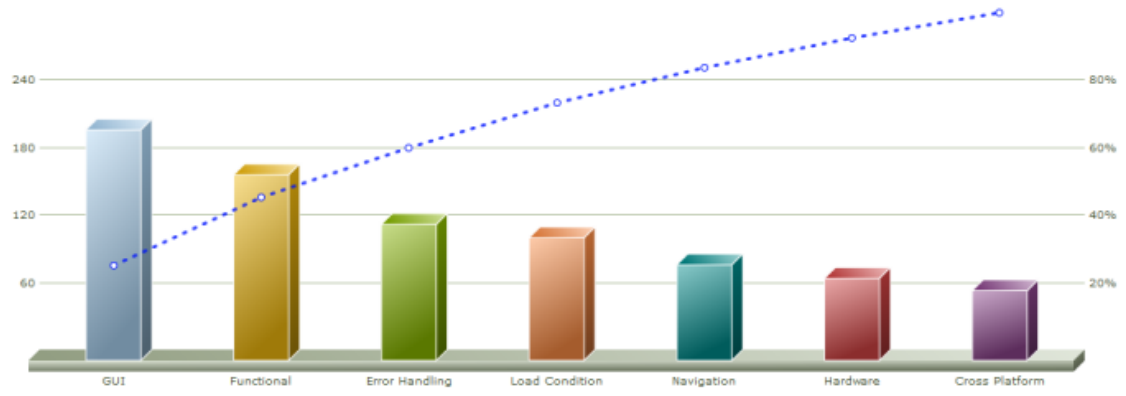
4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Pareto Chart

Pareto charts show both the values for a single numeric dimension over a set of categories and the relative contribution of each to the sum total. This combination provides an easy 80/20 view to identify key causes or factors and ordered from largest to smallest. Values display as ordered columns, from largest to smallest, while the relative contribution appears as a line.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies each column across the X axis. This is the *category* field. ■ One field with numeric data that you add as a *series*. This value is shown both a column heights and in the aggregate calculation used to define the contribution line.
Mobile Views	Yes

Configure This View

Choose the Pareto Chart view and:

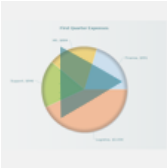
1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains labels for the chart columns and drop it in the **Label** field.
3. Drag the title of the column that contains numeric data into the **Value** field.
4. Optionally, drag the title of a column with text values to use as the **Tooltip** when the mouse passes over each column. If you omit this, tooltips simply do not appear in the chart.
5. Optionally, change the **Chart Visualization** property to set this as two dimensional (2D) or three dimensional (3D).
6. If the values of the **Label** field are not unique, change the **Aggregation** function from `None` to the function you want to use to calculate the value to use for each column.
7. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
8. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
9. Preview the chart at any time.

10. Once you are satisfied, save the view:
    a. Click **Finish**.
    b. Give the view a name, and optional description.
    c. Click **Save**.

Pie Chart



Pie charts display a single numeric data point for multiple records as percentages of a whole. The percentage is rendered as a "slice of pie" covering an arc within a circle. Pie charts are also similar to Doughnut Charts.

There are two possible relationships between a slice and the records in the data set:

- *1-to-1* where each record represents one slice.
- *Aggregated* where an aggregate value of multiple records are shown as one slice.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what each slice is. In the example shown above, the Department field is the label. For 1-to-1 relationships, each record represents one slice and the value of the label should be unique in each record. With aggregate relationships, multiple records are aggregated into a slice and the value of the label may be repeated in different records. Each unique label value defines one slice. ■ One field with numeric data that defines the value (and size) of each slice. For 1-to-1 relationships, the value for each record is the value of the slice. With aggregate relationships, all the values of the records that match one unique label value are used to calculate

	the value of the slice based on the aggregate function you choose.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Pie Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the pie slices and drop this in the **Label** field.

   Tip: For better visual results, the total number of pie slices should be less than 75.

3. Drag the title of one column that contains numeric data into the **Value** field.

   Tip: For best results, the values for slices should be at least 3% or larger of the total.

4. Optionally, drag the title of a column with text values to use as the **Tooltip** when the mouse passes over each pie slice. If you omit this, tooltips simply do not appear in the chart.
5. Optionally, change the **Chart Visualization** property to set this as two dimensional (2D) or three dimensional (3D).
6. If the values of the **Label** field are not unique, change the **Aggregation** function from `None` to the function you want to use to calculate the value to use for each slice.
7. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

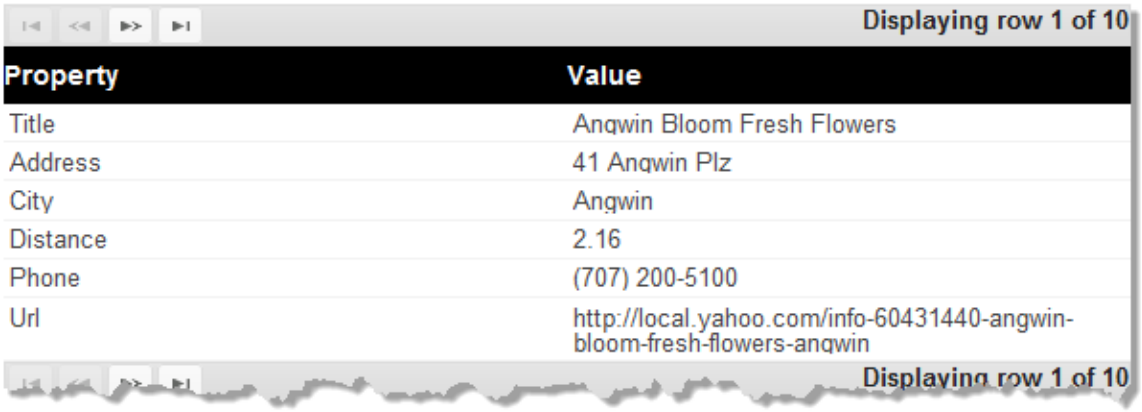
   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
8. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
9. Preview the chart at any time.

10. Once you are satisfied, save the view:
    a. Click **Finish**.
    b. Give the view a name, and optional description.
    c. Click **Save**.

Record Details View

This view display details from results for one record at a time in a two-column table. The first column shows each field heading for that record and the second column shows the values for the fields. This view automatically paginates data, allowing users to page through each record, one at a time.

You can use view configuration to add, remove or move columns, change headings, width, and other text formatting, apply different formats to columns based on the type of data, combine data into one column or apply formatting to define *key performance indicators* to columns or rows based on conditions you define.



Characteristics

Data Requirements	A set of repeating items with one or more fields.
Published Events	■ `rowclick` = user clicks in a cell to select a data row in the table ■ `headerclick` = user clicks in a cell in the column header row
Mobile Views	Yes

Configure This View

MashZone NextGen defines a default *dataset*, the path to the repeating items in your data to use for each row of the table, in the **Data Configuration** step. You can choose a different dataset, if needed.

Then define which columns to use in this table, the data that should appear in each column and the formats to apply to each column.

> **Note:** Configuration is almost identical to configuration for the "Grid and KPIs View" on page 946. With Property Sheets, there is no grouping as each rows displays on a separate page. Because there are only two columns, you cannot change the column width.

You can:

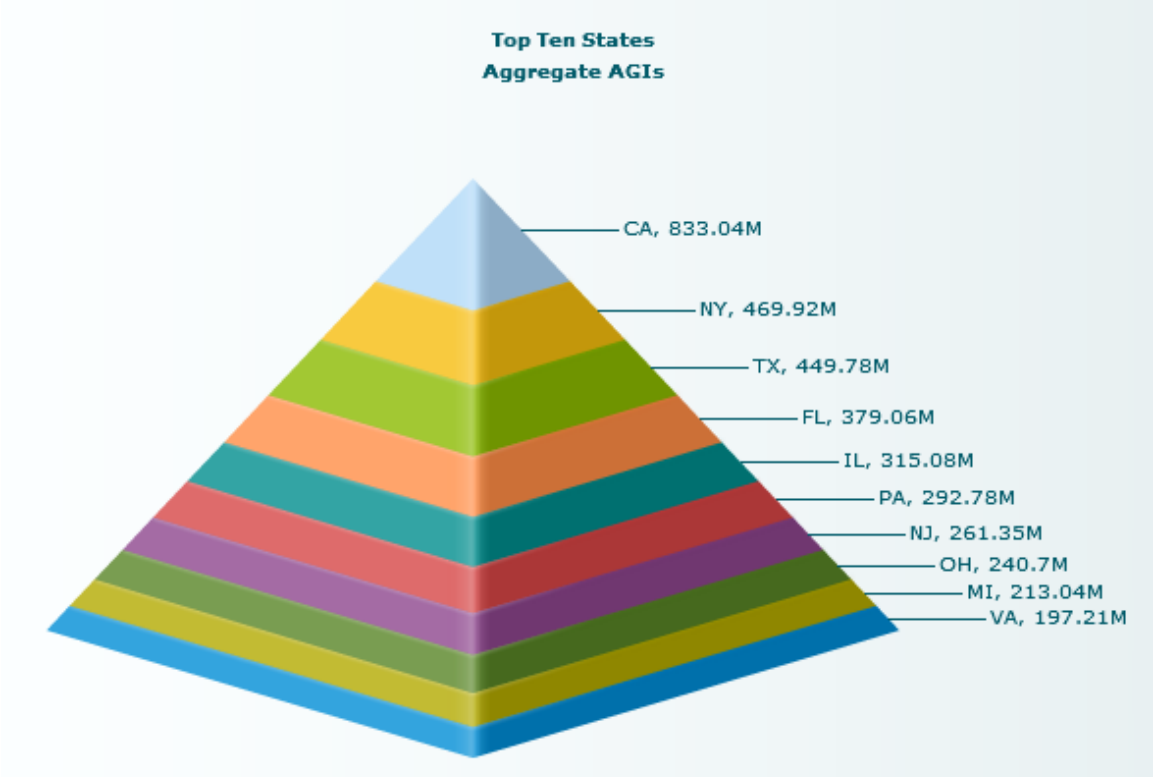
- Move or Remove Columns
- Add Columns and Assign Values
- Change Column Headings, Width and Text Formatting
- Show As: Change the Column Format for Different Types Data
- Use Templates to Combine Data, Add Literal Text or Add HTML Tags
- Add Key Performance Indicators (KPI)

Preview the results and once you are satisfied, save the view.

Pyramid Chart

Pyramid charts display a single series or data point for a set of records in a progression of slices of a three-dimensional pyramid. Both the decreasing circumference of the pyramid and the overall depth of each slice indicate the size of the data.

For more information on this view, see "Characteristics" on page 990 and "Configure This View" on page 991. For an alternate presentation of progressive numerical data, see "Funnel Chart" on page 934.



There are two possible relationships between a slice and the records in the data set:

- *1-to-1* where each record represents one slice.
- *Aggregated* where an aggregate value of multiple records are shown as one slice.

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies each slice. In the example shown above, the State field provides the label. For 1-to-1 relationships, each record represents one slice and the value of the label should be unique in each record. With aggregate relationships, multiple records are aggregated into a slice and the value of the label may be repeated in different records. Each unique label value defines one slice. ■ One field with numeric data that defines the value (and size) of each slice. Values shuld have some variation from

	large to small. In the example shown above, the aggregate AGI is the value. For 1-to-1 relationships, the value for each record is the value of the slice. With aggregate relationships, all the values of the records that match one unique label value are used to calculate the value of the slice based on the aggregate function you choose. For best results, the value for each record or aggregate value for each slice should be unique to ensure that slices do not misrepresent the data.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the Pyramid view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for each slice and drop this in the **Label** field.

   Tip: For best results, limit the number of categories to 15 or less.

3. Drag title of the column that contains numeric data into the **Value** field.
4. Optionally, drag and drop a column with text values in the **Tooltip** field.
5. If the values of the **Label** field are not unique, change the **Aggregation** function from `None` to the function you want to use to calculate the value to use for each slice.
6. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

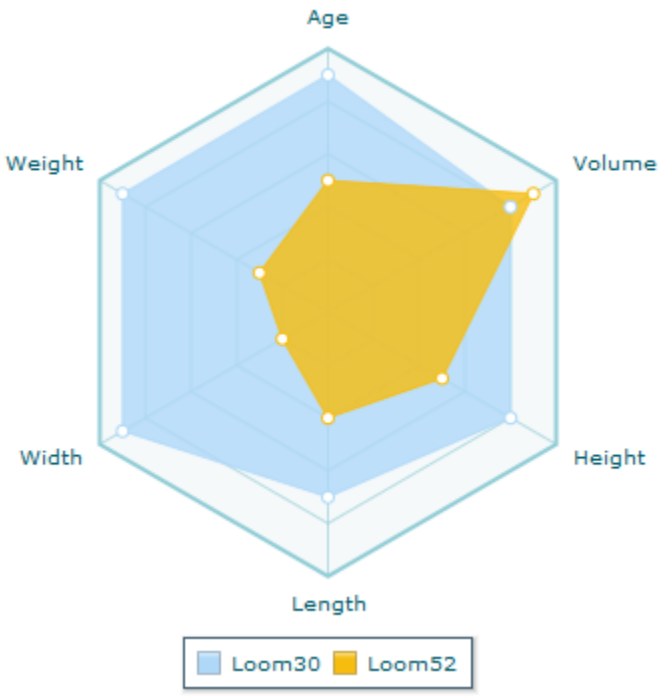
     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
7. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.

8. Preview the chart at any time.
9. Once you are satisfied, save the view:
    a. Click **Finish**.
    b. Give the view a name, and optional description.
    c. Click **Save**.

Radar Chart

Radar charts display irregular polygons plotted from a shared center with each polygon point representing the magnitude of a dimension of one item. Radar charts facilitate easy comparison of multiple dimensions for dataset rows. This is also sometimes called a spider chart.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what dimension is plotted on each vertix . This is the *category* field. ■ At least two fields with numeric data. Each field that you add as a *series* defines a vertix within the radar plot.

Mobile Views	Yes

Configure This View

Choose the Radar Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for each vertix and drop this in the **Category** field.
3. Drag the title of at least two columns with numeric data into the **Series** field. Each series adds a vertix to the plot.

   Tip: For best results, limit the number of series to 15 or less.

   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, Change Series Labels to make the chart easier to understand.
4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also Include Dynamic Content in Captions and Labels if desired.
5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Scatter Chart

A scatter chart displays two or more sets of numeric data points plotted as discrete points across a grid. Scatter charts allow users to determine a pattern, if any, in the data and the scope or distribution of the data. Scatter charts are related to Line Charts and Bubble Charts.

Tip: For best results with scatter charts, limit the number of repeating items to be plotted to 30 or less.

You can plot more than one set of points in a scatter chart. See "Characteristics" on page 994 and "Configure This View" on page 994 for more information.

Characteristics

Data Requirements	A set of repeating items with: ■ At least two fields with numeric data to use as the X and Y coordinates of the points to plot in the chart. ■ An alphanumeric field to use as a tooltip for each point.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the Scatter Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Complete one **Series**:
   a. Drag and drop the titles of columns with numeric data to both the **X Column** and **Y Column** fields.

      If needed, Change Series Labels for the X column to make the chart easier to understand.

b. Drag and drop the title of a column with text or numeric data into the **Tooltip** field.

3. To plot additional points in the scatter chart, click **Add series columns**. This collapses the fields for the previous Series and opens fields for another series. Complete the fields (see the previous stepfor instructions) for this new series.

> **Tip:** For best results, the numeric data for all of the series that you include in a scatter chart should fall within a similar range of values. One series with very large or very small values can skew the scale of the chart, making it difficult to see or understand other data.

   To re-open a Series, click the label for the series.

4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.

5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.

6. Preview the chart at any time.

7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Sparkline

Sparklines are useful in displaying trends for one or more data points (series) in a summary, highly compressed form with minimal highlights. Each series (column) displays one line progressing in the order that records appear in the dataset. Highlights are shown for the beginning, ending, high and low values.

Tip: The number of records determines how many points appear on each line. For large numbers of records (> 50), you can use pagination with this chart to provide better visibility in fluctuations.

For more information on using the Sparkline view, see "Characteristics" on page 996 and "Configure This View" on page 996.

For alternate formats, see "Sparkline Column" on page 997 and "Sparkline win_loss" on page 999. You can also include sparkline *mini-charts* for a column in a Grid and KPIs View. For a reverse relationship between records and data points, see "Line Chart" on page 972 or "Zoom Line Chart" on page 1023.

Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each field with numeric data that you add as a *series* defines a line in the chart. In the example shown above, there are five series for High, Low, Volume, Open and LastClose.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the SparkLine view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag at least one title for a column that contains numeric data into the **Series** field. Each column you add to Series adds a line to the chart.

   Tip: For best results, limit the number of series (the number of lines within the chart) to 8 or less.

   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.
3. For each series, set these properties:
   - Set a **Title**, and optionally a **Subtitle** for the line. Titles appear as a label at the beginning of the line.
   - If needed, add a **Number Prefix** or **Number Sufix** for the numeric labels for this line.

Common examples might be `$` as the prefix or `%` as the suffix.

4. Click **Next** and optionally set a **Caption** for the chart.

   You can also Include Dynamic Content in Captions and Labels if desired.

5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.

6. Preview the chart at any time.

7. Once you are satisfied, save the view:

   a. Click **Finish**.

   b. Give the view a name, and optional description.

   c. Click **Save**.

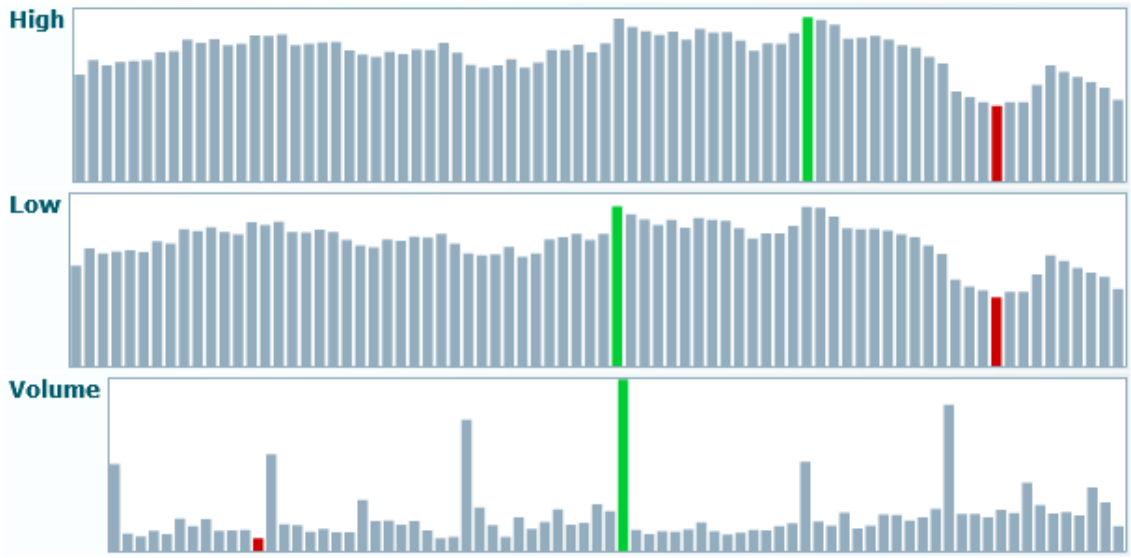
Sparkline Column

Sparkline column charts are useful in displaying trends for one or more *series* (a column in the dataset) in a summary, highly compressed form with minimal highlights. Each series displays a set of columns progressing in the order that records appear in the dataset. Highlights are shown for the high and low values.

> **Tip:** The number of records determines how many bars appear on each line. For large numbers of records (> 50), you can use pagination with this chart to provide better visibility for bars.

For more information on using the Sparkline Column view, see "Characteristics" on page 998 and "Configure This View" on page 998.

For alternate formats, see "Sparkline" on page 995 and "Sparkline win_loss" on page 999. You can also include sparkline column *mini-charts* for a column in a "Grid and KPIs View" on page 946. For a reverse relationship between records and data points, see the "Column Chart" on page 926 or "Bar Chart" on page 914.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each field with numeric data that you add as a *series* defines an X/Y axis for bars in the chart. In the example shown above, there are three series for High, Low and Volume.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the SparkLine Columns view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds an X/Y axis to the chart.

> **Tip:** For best results, limit the number of series to 8 or less.

It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

3. For each series, set these properties:
   - Set a **Title**, and optionally a **Subtitle** for the X/Y axis. Titles appear as a label to the left of the X/Y axis.
   - If needed, add a **Number Prefix** or **Number Sufix** for the numeric labels for this line.

     Common examples might be `$` as the prefix or `%` as the suffix.
4. Click **Next** and optionally set a **Caption** for the chart.

   You can also Include Dynamic Content in Captions and Labels if desired.
5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Sparkline win_loss

A Sparkline win_loss chart useful in displaying trends with respect to an origin line for one or more *series* (a column in the dataset) in a summary, highly compressed form with minimal highlights. One typical use is to display data points as win/loss/draw or profit/loss/break-even.

Each series displays a set of columns with respect to the origin line progressing in the order that records appear in the dataset. Highlights are shown for the number of *wins* (positive values), the number of *draws* (zeroes) and the number of *losses* (negative values).

Tip: The number of records determines how many bars appear on each line. For large numbers of records (> 50), you can use pagination with this chart to provide better visibility to bars.

For more information on using the Sparkline win_loss view, see "Characteristics" on page 1000 and "Configure This View" on page 1000. For alternate formats, see "Sparkline Column" on page 997 and "Sparkline Column" on page 997. You can also include sparkline win_loss *mini-charts* for a column in a Grid and KPIs View.

Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each field with numeric data that you add as a *series* defines an X/Y axis with origin line for bars in the chart. In the example shown above, there are five series for Income Tax, Property Tax, Sales Tax, Business Taxes and Other Taxes.
Published Events	None
Mobile Views	Yes

Configure This View

Choose the SparkLine win_loss view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds an X/Y axis with origin line to the chart.

   Tip: For best results, limit the number of series to 8 or less.

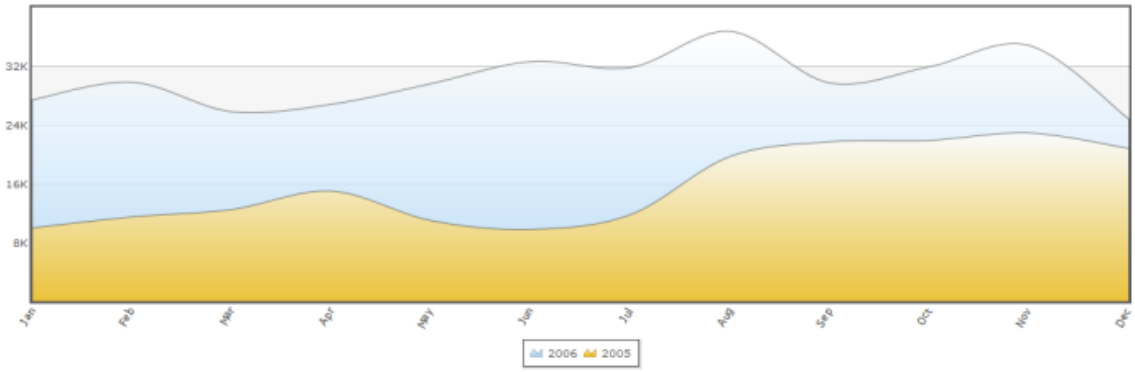
   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.
3. For each series, set a **Title**, and optionally a **Subtitle** for the X/Y axis. Titles appear as a label to the left of the origin line.
4. Click **Next** and optionally set a **Caption** for the chart.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.

7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Spline Area Chart

Spline Area charts display one or more numeric dimensions for a set of records as curved lines with the area underneath each curve filled with color. This is similar to area charts, but the fitted curves are more appropriate for continuous data.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies each tick on the X axis. This is the *category* field. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines a curved line in the chart.
Mobile Views	Yes

Configure This View

Choose the Spline Area view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

> **Tip:** With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a set of bars in a different color to the chart.

   > **Tip:** For best results, limit the number of series to 12 or less. It is also best if the numeric data are distinctly different but in a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, Change Series Labels to make the chart easier to understand.

4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

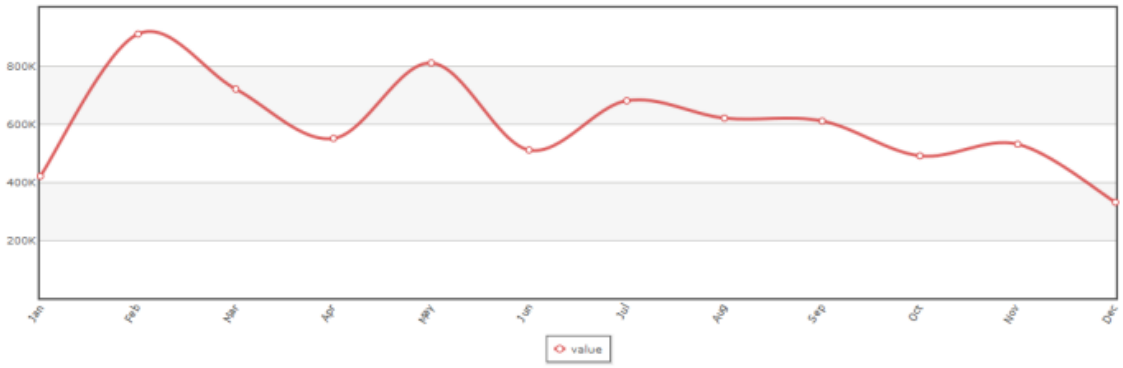
     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.

5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Spline Line Chart

Spline charts display one or more numeric dimensions for a set of records as curved lines that illustrate the trend or change between data points. This is similar to line charts, but the fitted curves are more appropriate for continuous data.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies each tick of the X axis. This is the *category* field. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines a curved line.
Mobile Views	Yes

Configure This View

Choose the Spline Line view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

   Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a curved line to the chart.

   Tip: For best results, limit the number of series to 12 or less.

   It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

If needed, Change Series Labels to make the chart easier to understand.

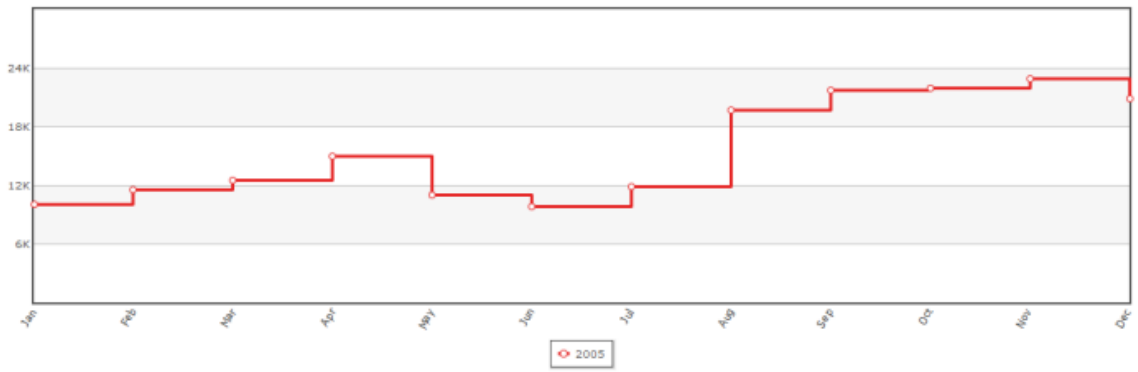
4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also "Include Dynamic Content in Captions and Labels" on page 906 if desired.
5. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Step Line

Step Line charts display lines that are vertical or horizontal only, resulting in *steps* that highlight changes in magnitude. Step lines are more appropriate for discrete data or for data where specific thresholds should be visible.



Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies each tick of the X axis. This is the *category* field.

	■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines a line.
Mobile Views	Yes

Configure This View

Choose the Step Line Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

   Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each Series you add adds a line to the chart.

   Tip: For best results, limit the number of series to 12 or less. It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

   If needed, Change Series Labels to make the chart easier to understand.
4. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also Include Dynamic Content in Captions and Labels if desired.
5. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.
6. Preview the chart at any time.
7. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Template View

This desktop-only view allows MashZone NextGen developers to create custom views for specific mashable information sources or mashups.

Note: This view is a beta feature.

MashZone NextGen developers can also create custom views that plug into the MashZone NextGen View Gallery so that they can be used with many mashables or mashups. See "Creating Pluggable Views or Libraries" on page 1045 for more information.

Custom views and templates combine HTML code with *template syntax* to map result fields to the appropriate locations within the HTML markup. This uses some aspects of the jQuery templating syntax. You can loop through repeating structures, handle conditional logic to generate appropriate HTML, display results that contain HTML markup and merge result data to produce the custom view.

Note: Custom views created with the Template view have two limitations:

- Custom views are not supported for mobile devices. The view can only be shown in desktop browsers.
- You cannot define topics or events that can be used to wire apps together in workspaces.

Users should be familiar with HTML and optionally CSS and JavaScript to use the Template view to create a custom view for an artifact. See the following sections for instructions and examples:

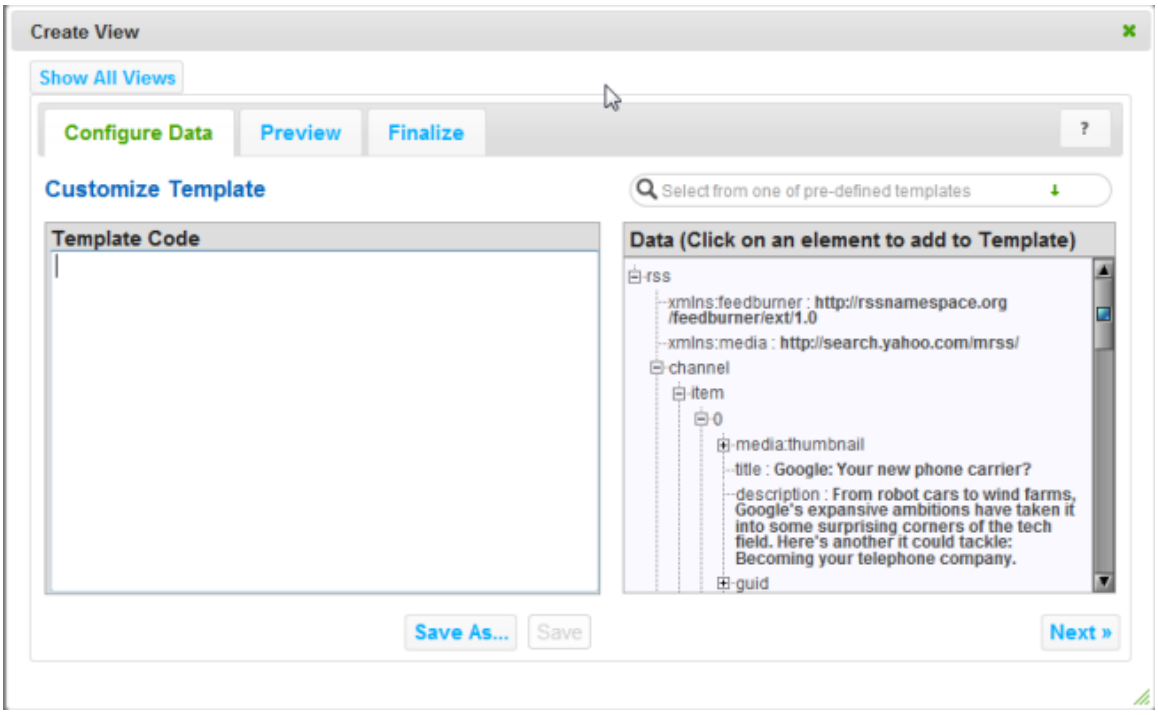
- "Create a Custom View" on page 1006
- "Map Result Fields to the Template" on page 1008
- "Map Results Fields That Contain HTML Markup" on page 1009
- "Iterate Through Repeating Results" on page 1011
- "Add Conditional Logic" on page 1012
- "Use CSS in Template Code" on page 1013
- "Use Images in Template Code" on page 1015
- "Use JavaScript in Template Code" on page 1015
- "Examples" on page 1017

Create a Custom View

You can create a template directly, by adding HTML tags, mapping result fields and adding other template syntax until you have the custom view you want. Or you can

select a global view template as a starting point and tweak the HTML and template syntax as needed.

- In the Configure Data step, type HTML tags directly in the **Template Code** pane. The template code should be well-formed, wrapped entirely in one tag, such as .



  Then update the code for the template as needed:

- Map Result Fields to the Template or Map Results Fields That Contain HTML Markup to define where result data should appear.
- If needed, Iterate Through Repeating Results,
- Add Conditional Logic,
- Use CSS in Template Code,
- Use Images in Template Code or
- Use JavaScript in Template Code to define the final output.
- Use **Preview** at any time to preview the results of your code.
- Once you are satisfied, click **Save as**.
- Enter a name and optional description for this custom view. Leave the Template Scope set to **Specific**.
- Click **Save**.

Map Result Fields to the Template

Mapping result fields defines where data will appear within the HTML code for this view or template. You use the jQuery syntax for variable substitution to map fields:

```
${path.to.datanode}
```

See jQuery Template Syntax for Variable Substitution for more details on this syntax.

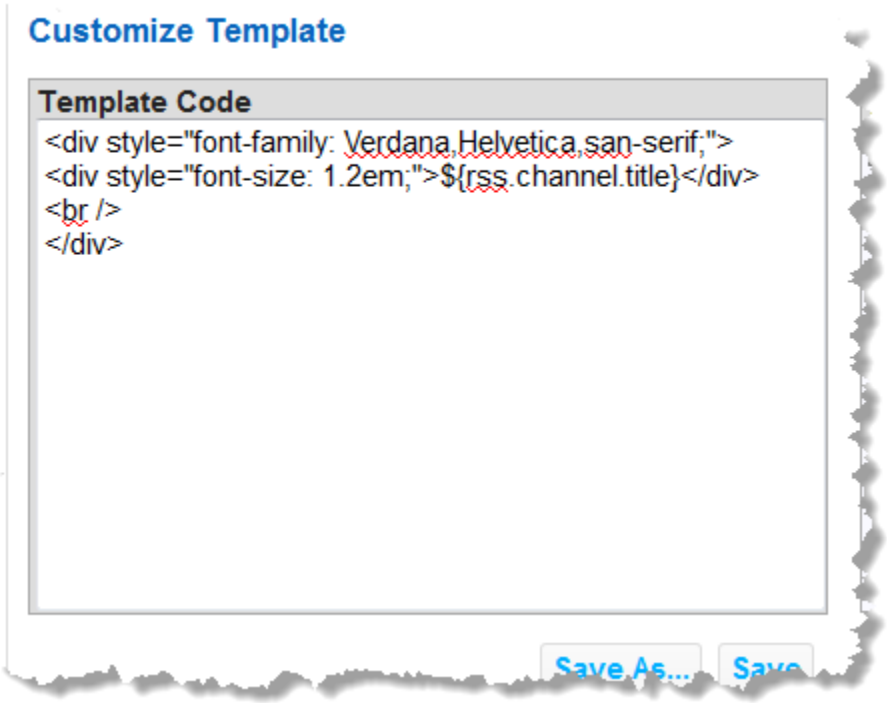
You should only map fields that contain data. If you map a complex node (an object) by mistake, Preview shows `[object Object]` as the value.

Rather than entering mapping syntax directly, you can automatically generate this syntax using the Data pane in the Configure Data step:

- **Data Field Not in a Repeating Item**

  1. In the Template pane, type `${}` inside the HTML element or attribute that should contain the data for the field.
  2. Place the cursor inside the braces.
  3. Expand the results tree in the Data pane.
  4. Click the field in the Data pane that you want to map to the current position in the template.

  The template code is updated with the full path to this field (`${full.path.to.field}`):



- **Data Field in a Repeating Item**

  1. Place the cursor inside the HTML element or attribute where you want the data to appear.
  2. Expand the results tree in the Data pane.
  3. Click the field from one repeating item in the Data pane that you want to map to the template.

  The template code is updated with the mapping syntax and the relative path to this field, starting from the repeating item node. For example: `${description}` for one field inside `rss.channel.item`.

  This syntax assumes that you are mapping data fields for repeating items within a loop, where the path to the repeating item node is already defined. See "Iterate Through Repeating Results" on page 1011 or "Example 133. Repeating Items" on page 1017 for examples.

- **Data Field in a Specific Repeating Item Instance**:

  1. Place the cursor inside the HTML element or attribute where you want the data to appear.
  2. Expand the results tree in the Data pane.
  3. Click the field from one repeating item in the Data pane that you want to map to the template.

     This adds the mapping syntax using a relative path to this field, such as `${title}`.

  4. Add the full path to this field, including the specific index number for the repeating instance that you want. For example: `${rss.channel.item[0].title}` to map the title of the first item in an RSS feed.

Map Results Fields That Contain HTML Markup

Some result fields contain HTML markup along with data. Common examples include fields in RSS or Atom web feeds. If you map these result fields using the basic mapping syntax, `${`*path.to.fieldname* `}`, the view shows the HTML tags for this field such as this example:

To handle result fields that contain HTML, use the {{html *path.to.fieldname*}} syntax instead.

Note: See jQuery Template Syntax for HTML Evaluation for more details on this syntax.

1. Enter `{{html }}` within the HTML tags where this field should appear.
2. Place the cursor after the space in the html statement and before the end braces (`}}`).
3. Expand the results tree in the Data pane.
4. Click the field in the Data pane that has HTML markup that you want to map to the current position in the template.

   The mapping syntax that this generates depends on whether this is a single field or a field inside repeating items. For single fields, the syntax is generated correctly with a full path to that field, such as `{{html rss.channel.title}}`.

   For fields in repeating items, the syntax that is generated has a relative path, which is correct, but the path uses the standard mapping syntax such as `{{html ${description}}}`. Preview will show an error `invalid property id` rather than rendering the view.
5. If needed, remove the `${` and `}` symbols that were generated around the path to this field.

With the correct syntax, the view now renders the HTML code from this field:

Iterate Through Repeating Results

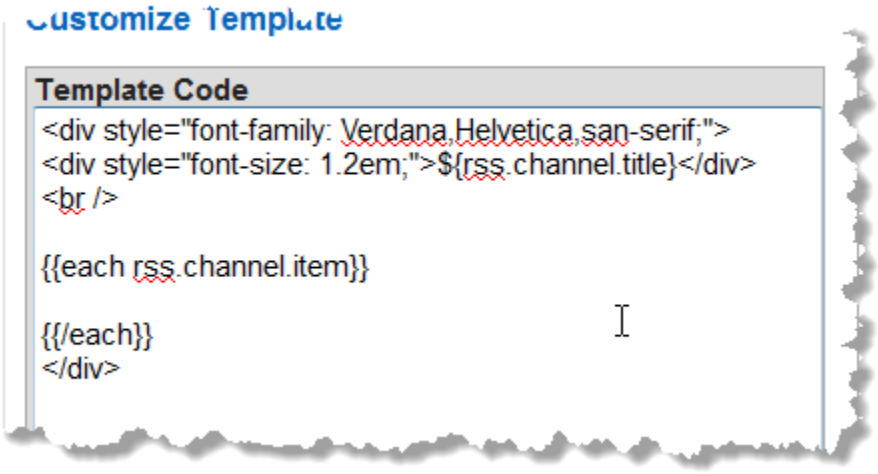
In many cases, the results contain a repeating structure with data that the view should iterate through, generating HTML and data for each instance. Use the {{each *structure.path* }} template syntax to performs loops in a template:

1. Enter the start- and end-loop statements where you want to begin looping in the view:

```
{{each }}
{{/each}}
```

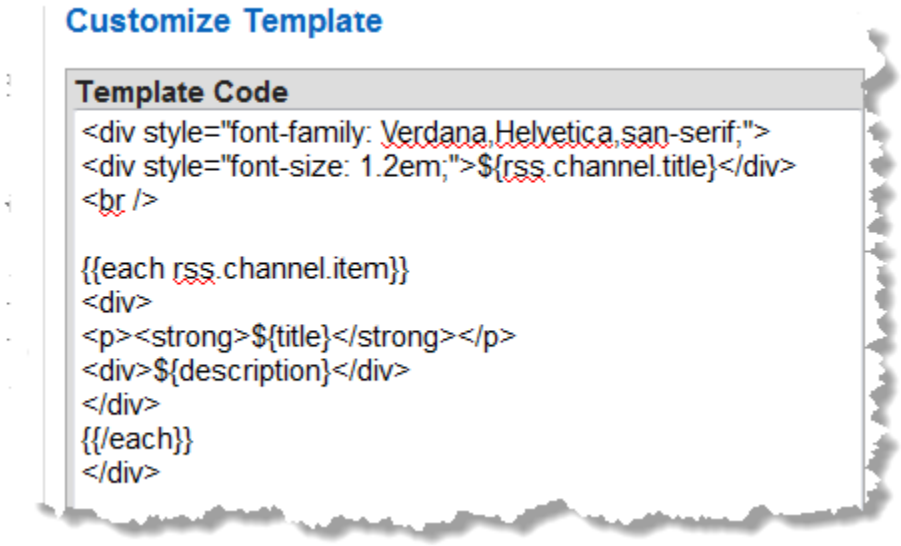
Note: See jQuery Template Syntax for Loops for more details on looping syntax.

2. Place the cursor after the space in the start-loop statement.
3. Click in the Data pane on the repeating node that should be used as the iterator for the loop:



This adds the full path to the repeating node for the loop.

4. Within the start- and end-loop statements, add the HTML that should be generated for each loop iteration. Map individual fields from the repeating items as needed (see "Map Result Fields to the Template" on page 1008).



Mapping uses relative paths to fields within each repeating item, as shown above. See also "Example 133. Repeating Items" on page 1017 for a more detailed example.

Add Conditional Logic

You may also need templates or custom views to have conditional logic or handle different cases in the results. To add conditions, use these template statements:

- `{{if` *condition* `}}...{{/if}}`: where the condition is any valid JavaScript condition. To check for the presence of a field in the results, use the path to that field as the condition.
- `{{else` *condition* `}}...{{/else}}`: can be used within an if-statement. The condition is any valid JavaScript condition. To check for the presence of a field in the results, use the path to that field as the condition.
- `{{else}}...{{/else}}`: can be used within an if-statement to provide a default behavior if none of the conditions are true.

To generate the path to a field as part of a condition

1. Place the cursor inside the `if` or `else` start-statement
2. Add a space after `if` or `else`.
3. Click on the field you want in the Data pane.

Insert the appropriate HTML and other template syntax within the start- and end-statements.

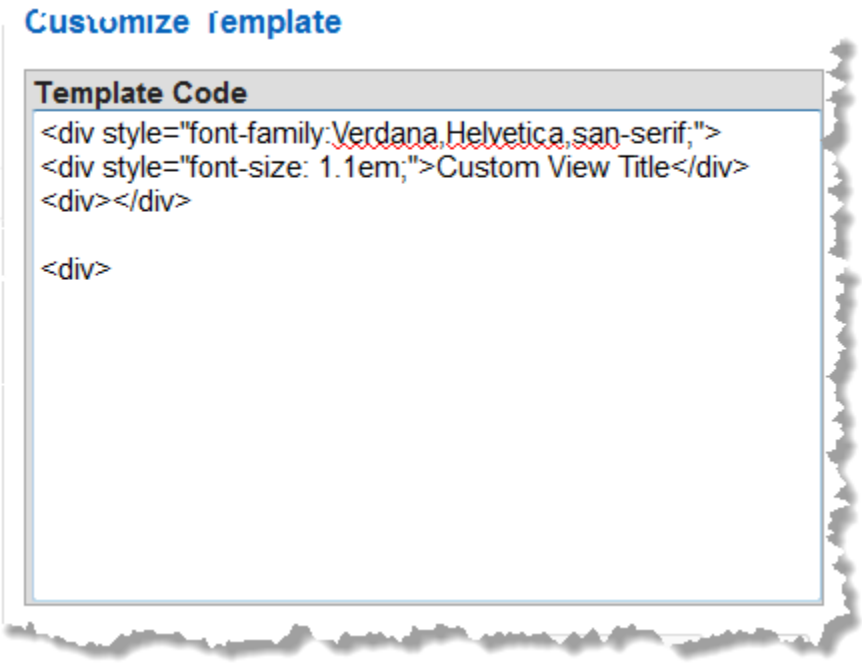
This example change the background color and text color based on the value of a field:

```
{{if PercentChange > 0}}
<div style="width: 90px;padding: 5px; border: 1px solid #ccc; margin: 5px;float: left;
      background-color: #5ab000;color:#fff;">
{{else PercentChange < -3}}
<div style="width: 90px;padding: 5px; border: 1px solid #ccc; margin: 5px;float: left;
     background-color: #c11;color:#fff;">
{{else}}
<div style="width: 90px;padding: 5px; border: 1px solid #ccc; margin: 5px;float: left;
     background-color: #ff9;color:#333;">
{{/if}}
```

See If Syntax and Else Syntax for more information on conditional statements in jQuery templates. See "Example 134. Conditional Logic, No Repeating Items and Namespaces" on page 1017 for a more detailed example.

Use CSS in Template Code

You can use CSS as local styles using the `style` attribute directly on any HTML element in a template. For example:



You can also use the `class` attribute to refer to CSS classes defined in external CSS stylesheets. You *must* include `<link>` elements in the template to link to any external CSS stylesheets.

External CSS stylesheets used in templates must either:

- Have a fully-qualified URL where the stylesheet is hosted. For example:

```
<div>
<link type="text/css" rel="stylesheet"
  href="http://www.jackbe.com/prestodocs/v3.0/prestodocs.css"/>
<div
>
<div
>View Title</div>
...
</div>
</div>
```

  Generally, you should use this option for external stylesheets hosted by third parties or custom stylesheets for your organization that are hosted *outside* of MashZone NextGen. Hosting custom stylesheets for your organization in the MashZone NextGen or MashZone NextGen Hub web applications is not recommended as this complicates deployment and upgrade migrations.

- Have been uploaded to MashZone NextGen and use a MashZone NextGen URL.

  This option is recommended for custom stylesheets for your organization or from third-parties when they cannot be hosted outside of MashZone NextGen or these resources should be available for use in any custom view.

Note: External resources can only be uploaded to MashZone NextGen by MashZone NextGen administrators.

This example links to a stylesheet that has been uploaded to MashZone NextGen. The full path for this URL is defined when the resource is uploaded, but is generally in the form `http:`*app-server:port* `/presto/files/`*filename* :

```
<div>
<link type="text/css" rel="stylesheet"
  href="http://localhost:8080/presto/files/myView.css"/>
<div
>
...
</div>
</div>
```

Use Images in Template Code

To use images in a custom view or template, they must be hosted outside of MashZone NextGen at a fully-qualified URL *or* a MashZone NextGen administrator must upload the image file to MashZone NextGen. Image files may be in any commonly-used format for web pages, such as GIF, PNG or JPEG.

You can add images to a custom view or template using CSS styles or using an <img> tag in the view or template HTML code. This is an example using CSS and an external image:

```
...
<div style="height: 180px; width: 250px;
  background:url('http://myOrg.com/images/logo.png"></div>
...
```

With images that have been uploaded to MashZone NextGen you simply use the MashZone NextGen URL. The full path for this URL is defined when the resource is uploaded, but is generally in the form `http:`*app-server:port* `/presto/files/`*filename* . This example uses an <img> tag and a MashZone NextGen URL:

```
...
<img src="http://localhost:8080/presto/files/myHeader.png"/>
...
```

Use JavaScript in Template Code

You can add JavaScript code directly to a custom view or template using the <script> tag. For example:

```
<div>
<div id="myMsg"></div>
<script>jQuery("myMsg").html("Welcome stranger!");</script>
</div>
```

You can also refer to external JavaScript libraries. External libraries must:

- Have a fully-qualified URL where the library is hosted. For example:

  ```
  <div>
  <script type="text/javascript"
    scr="http://some.example.com/libraries/commonView.js"/>
  ...
  ```

```
</div>
```

Generally, you should use this option for external libraries hosted by third parties or libraries for your organization that are hosted *outside* of MashZone NextGen. Hosting JavaScript libraries for your organization in the MashZone NextGen or MashZone NextGen Hub web applications is not recommended as this complicates deployment and upgrade migrations.

- Have been uploaded to MashZone NextGen and use a MashZone NextGen URL.

  This option is recommended when JavaScript libraries cannot be hosted outside of MashZone NextGen or these resources should be available for use in many custom views.

  Note: External resources can only be uploaded to MashZone NextGen by MashZone NextGen administrators.

  This example uses a JavaScript library that has been uploaded to MashZone NextGen. The full path for this URL is defined when the resource is uploaded, but is generally in the form `http:`*app-server:port* `/presto/files/`*library-name*:

```
<div>
<script type="text/javascript"
  scr="http://localhost:8080/presto/files/commonView.js"/>
...
</div>
```

Loading and Scope

Custom views become part of the user interface for apps which in turn are published and run inside of other web pages or mobile devices. Thus scripts are loaded and run within the body of another page. Typically, the DOM for the page is fully loaded.

Scripts should also use a unique namespace to ensure there are no naming conflicts with other code running within that page.

Events and Event Handlers

Custom views support all of the standard DOM events. Developers can also use jQuery functions to register event handlers or to define and fire custom events. For example:

```
...
<button id="event1">Trigger an event</button>
<script type="text/javascript">
  jQuery("#event1").click(function(){
    jQuery(this).parent().trigger("myEvent", {
      eventdata: { type: 'button', time: (new.Date()).toString() }
      }
    );
  });
</script>
...
```

Examples

More detailed examples are included for: Example 133. Repeating Items, Example 134. Conditional Logic, No Repeating Items and Namespaces and Example 135. Linked CSS and JavaScript.

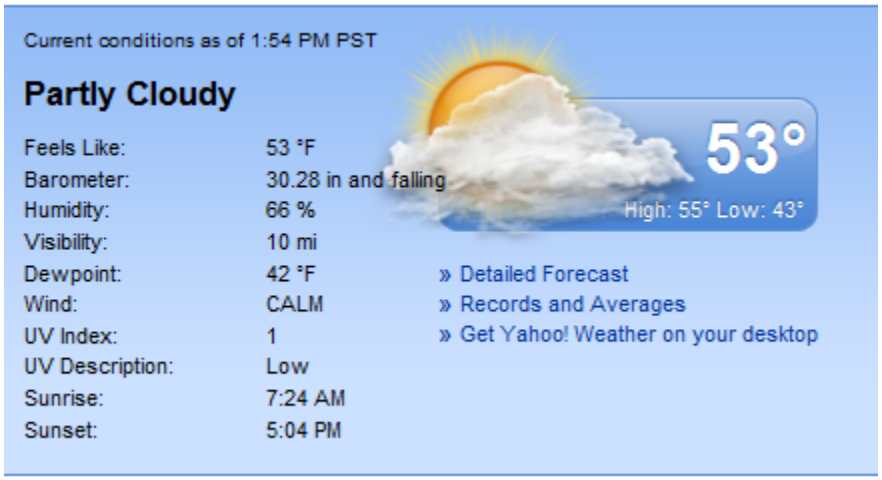
Repeating Items

This example shows a simple layout for a set of repeating record nodes:

```
<div>
  {{each records.record}}
    <div style="padding: 5px;border-bottom: 1px solid #eee;">
      <div style="float:left; width: 60%;">
        <h4>${offense}</h4>
        <p><em>${city}: Reported at ${lastmodifieddate}</em></p>
        <p>${narrative}</p>
        <p>District: ${district}</p>
        <p>Address: ${siteaddress}</p>
      </div>
    </div>
  {{/each}}
</div>
```

Conditional Logic, No Repeating Items and Namespaces

This next example uses the Yahoo Weather syndicated feed to produce a view similar to the standard weather displayed by Yahoo:



This example handles a result that has no repeating items. Even though it is an RSS web feed, Yahoo returns a single item with weather information for the selected zip code. Some of the interesting points in this example include:

- Mapping to non-repeating nodes and nodes with namespaces:

```
<em>Current conditions at
    <strong>${rss.channel['yweather:location'].city}</strong>
    as of ${rss.channel.lastBuildDate}
  </em>
```

Nodes with namespaces use array syntax, such as ['yweather:location']. The path to this non-repeating node uses the full path from the root node of the result.

- A basic if/else example:

```
<dd style="position:relative;">
  ${rss.channel['yweather:atmosphere'].pressure} in and
    {{if rss.channel['yweather:atmosphere'].rising > 0}}
      rising
    {{else}}
      falling
    {{/if}}
 </dd>
```

- Using mapping and conditional template syntax within the value of an HTML attribute:

```
<div style="height: 180px; width: 250px;
background:url('http://l.yimg.com/a/i/us/nws/weather/gr/
${rss.channel.item['yweather:condition'].code}
{{if rss.channel.item['yweather:condition'].date.indexOf("pm") > 0}}
d
{{else}}
n
{{/if}}
.png');"></div>
```

The full code for this template is shown below:

```
<div style="height:auto;background-color: #CCE1FF;font-size:smaller; padding: 10px;">
  <em>Current conditions at
    <strong>${rss.channel['yweather:location'].city}</strong> as of
    ${rss.channel.lastBuildDate}
  </em>
  <h4>${rss.channel.item['yweather:condition'].text},
    ${rss.channel.item['yweather:condition'].temp}
    ${rss.channel['yweather:units'].temperature}
  </h4>
  <div>
    <div style="float:left; width: 30%;">
      <dl>
        <dt style="float: left; min-width: 100px;">Feels Like:</dt>
        <dd>${rss.channel.item['yweather:condition'].temp}</dd>
        <dt style="float: left; width: 100px;">Barometer:</dt>
        <dd style="position:relative;">
          ${rss.channel['yweather:atmosphere'].pressure} in and
            {{if rss.channel['yweather:atmosphere'].rising > 0}}
              rising
            {{else}}
              falling
            {{/if}}
        </dd>
        <dt style="float: left; width: 100px;">Humidity:</dt>
        <dd>${rss.channel['yweather:atmosphere'].humidity} %</dd>
        <dt style="float: left; width: 100px;">Visibility:</dt>
        <dd>${rss.channel['yweather:atmosphere'].visibility} mi</dd>
        <dt style="float: left; width: 100px;">Dewpoint:</dt>
        <dd>${rss.channel['yweather:wind'].chill} 5°F</dd>
        <dt style="float: left; width: 100px;">Wind:</dt>
        <dd>${rss.channel['yweather:wind'].speed} mph, chilll:
          ${rss.channel['yweather:wind'].chill}</dd>
        <dt style="float: left; width: 100px;">UV Index:</dt>
        <dd>--</dd>
```

```
        <dt style="float: left; width: 100px;"> UV Description:</dt>
        <dd>Low</dd>
        <dt style="float: left; width: 100px;">Sunrise:</dt>
        <dd>${rss.channel['yweather:astronomy'].sunrise}</dd>
        <dt style="float: left; width: 100px;">Sunset:</dt>
        <dd>${rss.channel['yweather:astronomy'].sunset}</dd>
      </dl>
    </div>
    <div style="float: left;">
      <div style="height: 180px; width: 250px;
           background:url('http://l.yimg.com/a/i/us/nws/weather/gr/${rss.channel.item['yweather:c
           {{if rss.channel.item['yweather:condition'].date.indexOf("pm") > 0}}d{{else}}n{{/if}}.
      </div>
    </div>
    <div
 />
  </div>
</div>
```

Linked CSS and JavaScript

This example uses sample Yahoo Weather Given Zip Code mashables and these CSS and JavaScript files which have been uploaded to MashZone NextGen using the default file names:

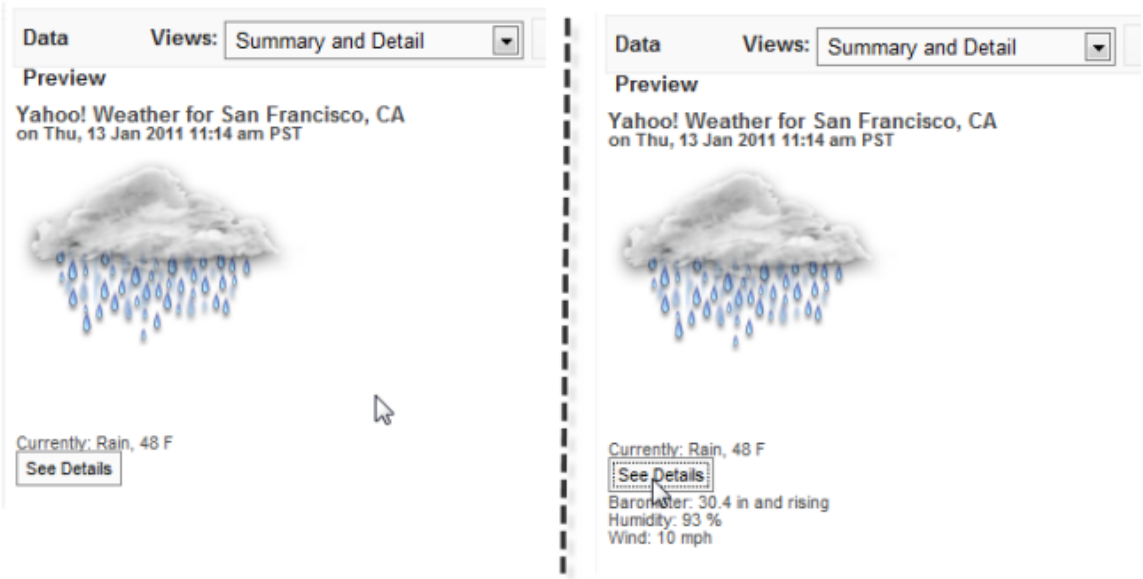
weathersumdetail.css	`.weather {font-siz`
	`.wtitle {font-size`
	`.wsubtitle {font-w`
	`.wsummary{ display`
	`.wdetail{ display:`
	`button{ margin-top`

weathersumdetail.js	`jQuery("button")`

The custom view code is:

```
<div
>
  <link type="text/css" rel="stylesheet"
    href="http://localhost:8080/presto/files/weathersumdetail.css"/>
  <script type="text/JavaScript"
    src="http://localhost:8080/presto/files/weathersumdetail.js"/>
  <div
>${rss.channel.description}</div>
  <div
>on ${rss.channel.lastBuildDate}</div>
  <img height="180px" width="250px"
       src="http://l.yimg.com/a/i/us/nws/weather/gr/${rss.channel.item['yweather:condition'].code
       {{if rss.channel.item['yweather:condition'].date.indexOf("pm") > 0}}d{{else}}n{{/if}}.png"
  <div
>Currently: ${rss.channel.item['yweather:condition'].text}, ${rss.channel.item['yweather:conditio
  <button>See Details</button>
  <div
>
    <div>Barometer: ${rss.channel['yweather:atmosphere'].pressure} in and
    {{if rss.channel['yweather:atmosphere'].rising > 0}}rising{{else}}falling{{/if}}</div>
    <div>Humidity: ${rss.channel['yweather:atmosphere'].humidity} %</div>
    <div>Wind: ${rss.channel['yweather:wind'].speed} mph</div>
  </div>
</div>
```

The JavaScript defines a handler for the button in the template that updates the CSS style for the with `wdetail` as a class. This displays further stats for the weather. Both the initial view and the update from clicking More Details are shown below:

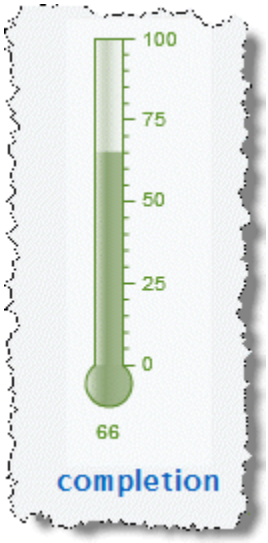


Thermometer Gauge

Thermometer gauges display a single numeric data point in relation to a scale that is rendered as a vertical thermometer. This can be literal temperature data or simply a scale to indicate importance, progress and so on.

> **Note:** Because a gauge shows a relation of one data point, gauge views create multiple pages, one for each datapoint, when mashable or mashup results contain repeating items. This is very similar to the Record Detail view where each row in the results is displayed as one page.

As with all MashZone NextGen built-in gauge views, you can configure several gauges per view with a mix of gauge types.



Characteristics

Data Requirements	A set of repeating items with at least one field with numeric data. Each row in the results creates a separate gauge.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

To configure a view with a thermometer gauge

1. Add a Gauge to This View.
2. Configure the Thermometer Gauge.
3. In **Customize View**, under Basic Options, enter the title for the entire view in the **Caption** property.

   You can also Include Dynamic Content in Captions and Labels if desired.
4. If desired, add other gauges to this view.

   Define how the gauges are stacked within the view in the **Stack Gauges** property under Basic Options in **Customize View**. Up to three gauges will generally stack nicely. More than three may add scroll bars to the view.
5. Preview the view and once you are satisfied, save the view.

Add a Gauge to This View

1. If needed, change the **Select Dataset** property to find the node that contains the field you want to display in the gauge. This is the *parent element*, not the actual field.

   Note: The parent you select can be a repeating element in your data, but unlike many other views, a repeating element is *not* required. If you select a repeating element, each repeating item generates another page with its own gauge.

2. Drag the title of the column that has the data you want to display in the gauge into the **Select Series** area.

   Important: *Each* column that you select as a series defines one gauge for this view. You can have up to three gauges in a view.

3. Choose the type of gauge you want and complete configuration properties.

Configure the Thermometer Gauge

1. If needed, select the Series column for the gauge and change the **Gauge Type** to `Thermometer`.

2. Define the scale for this gauge in the **Color Range**.

   The scale defaults to 0-100, or a maximum close to the maximum value for this column. Unlike some other types of gauges, intermediate numbers in the scale do *not* affect the fill color for the thermometer.

   Gauge scales can be any range of contiguous numbers, negative or positive and can use integers or decimals as needed. The scale must have at least the lowest and highest numbers that you want to show.

   You can add numbers or remove numbers from the scale.

   - To add a new number to the scale, enter a number and click **Add Value**.
   - To remove a number, click on the number in the scale.

3. Enter the title for *this* specific gauge as the **Title**.

4. If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

   Common examples might be `$` as the prefix or `%` as suffix.

5. Use **Customize View**, if needed, to change the colors shown in the gauge or update other visual properties to get the exact appearance you want.

   To change the colors shown in the gauge:

   a. Click **Show Advanced Options**.

   b. Expand the **Advanced Properties** section.

c. In **Color Range Palette**, click inside the color cell (white by default) to pick the first (or next) color for the dial.

d. Choose the color you want from the color picker and click **Set Color**. The color cell now shows the color you selected.

e. Click **Add Color** to add this color to the list of colors used in the dial

f. Add a color for each range of numbers you defined for the dial.

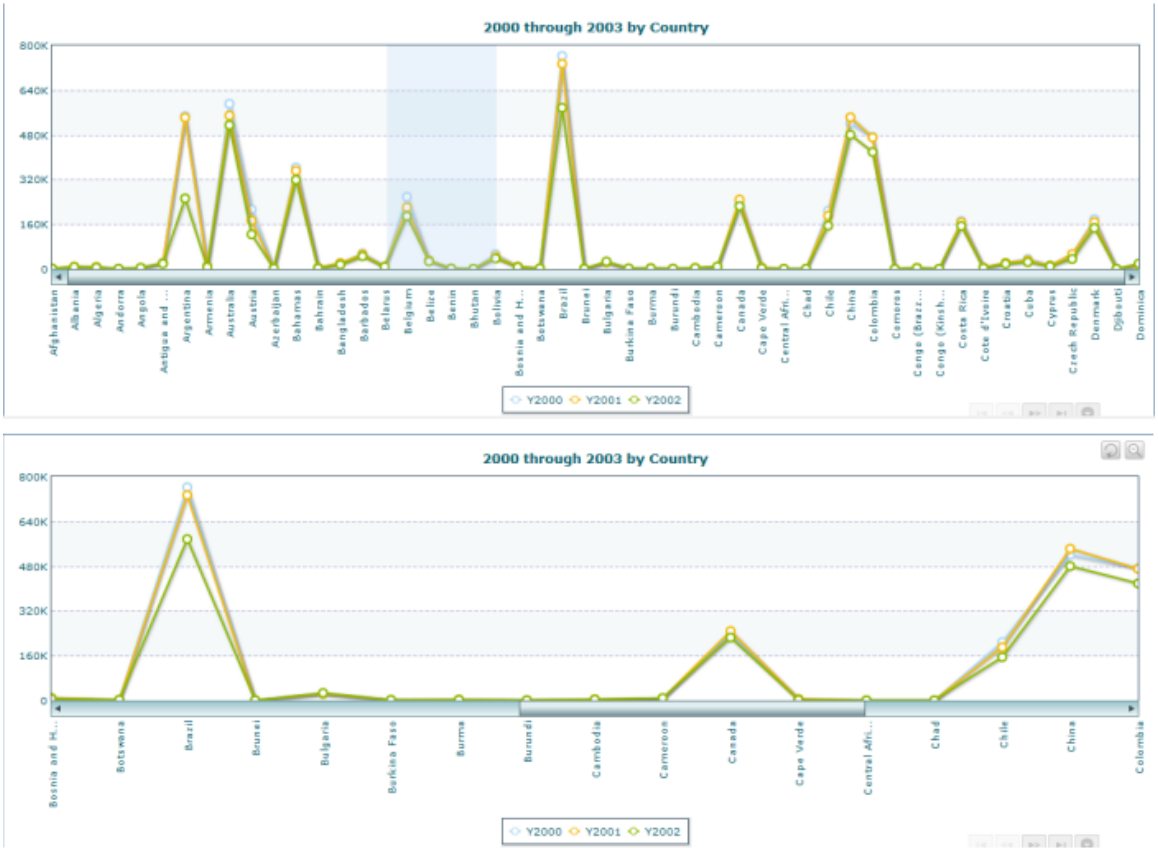
   If needed, simply click on a color in the list to delete that color.

6. Preview the gauge, as needed.

Zoom Line Chart

Zoom Line charts are s used for datasets with a large number of records to allow users to zoom in on specific areas of the chart to see details. Like line charts, zoom line charts display lines for numeric data, one for each series or column, for a set of records.

When the dataset has a large number of records, the number of records per page may need to be high which can obscure details, such as the top example shown below.



Users can simply select a portion of the X axis to zoom in and see details, as shown in the bottom example above. A small toolbar in the zoom view provides controls to return to the summary view:

Line charts are also related to Scatter Charts and Area Charts. For a reverse relationship between data points and records, see "Sparkline" on page 995. See Line Chart for moderate or small datasets. a

Characteristics

Data Requirements	A set of repeating items with: ■ One field with a label that identifies what is on each tick of the X axis. This is the *category* field. In the example shown above, the Month field is the category. ■ At least one field with numeric data. Each field with numeric data that you add as a *series* defines a line in one color.
Published Events	`element-click` = user clicks on any of the graphic objects or areas that represent data in this chart
Mobile Views	Yes

Configure This View

Choose the Zoom Line Chart view and:

1. If needed, change the **Select Dataset** field to find the repeating items that contain the fields you want to chart. This must be the *parent* node to the fields you want to use and it must be a repeating node.
2. Drag the title of the column that contains the labels for the X axis and drop this in the **Category** field.

   Tip: With large numbers of categories (greater than 10-12), use the Chart Scroll property for better visual results.

3. Drag the title of at least one column that contains numeric data into the **Series** field. Each column you add to Series adds a line in a different color to the chart.

   Tip: For best results, limit the number of series (the number of lines per category) to 12 or less.

It is also best if the numeric data are distinctly different but within a similar range of values. One very large or very small value can skew the chart scale, making it difficult to see or understand other data.

4. If desired, Add a Target Line to the chart.
5. Click **Next** and set basic options for the chart in the Customize View step:
   - Set a **Caption** or **Subcaption** for the chart or gauge.
   - Set captions, if needed, for the **X** and **Y** axes.
   - If needed, add a prefix or suffix for all numeric labels for this gauge or chart.

     Common examples might be `$` as the prefix or `%` as suffix.

   You can also Include Dynamic Content in Captions and Labels if desired.
6. If needed, click **Switch to advanced mode** to set other visual properties for the chart. See "Advanced Configuration Properties" on page 1025 for details.
7. Preview the chart at any time.

   Note: Preview defaults to show 20 records per page. Zoom line charts are typically used with large datasets and higher records per page.
8. Once you are satisfied, save the view:
   a. Click **Finish**.
   b. Give the view a name, and optional description.
   c. Click **Save**.

Advanced Configuration Properties

Following is a comprehensive list of advanced configuration properties for the MashZone NextGen built-in Chart, Gauge and Sparkline views. Not all properties are available for each view.

Important: Advanced configuration properties for charts *only* apply to desktop views. They do not affect phone or tablet views.

Basic configuration properties, such as **Caption**, apply to views for all devices (desktop, phone and tablet).

Background

Background Color	Both the primary, and optional secondary background colors for the chart. The default primary color is white.

	This property has both a sample color button that allows you to select a color and a field that lists the hexadecimal value(s) of the color(s) you have selected. If you want to use a gradient in the background, add a secondary color. Or delete the code for the primary color and add a different primary color.
Background Transparency	How opaque each background color is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Show Border	Determines whether there is a border around the entire chart. Defaults to true.
Border Color	The color of the chart border.
Border Transparency	How opaque the chart border is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Border Thickness	The thickness of the chart border in points.
Background Gradient Ratio	Ratio, as a percentage, for the primary and secondary colors in a background using a gradient.
Background Gradient Angle	Angle of the gradient of background colors in degrees from 0-360. Default is 180 (horizontal gradient).
Background Image	The URL to an image to use as the chart background.
Background Image Transparency	How opaque the image is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Background Image Display Mode	How the image fills the background: ■ `Stretch` ■ `Tile` ■ `Fit` ■ `Fill` ■ `Center` ■ `None`

Background Image Vertical Alignment	`Left`, `Right` or `Center`.
Background Image Horizontal Alignment	`Left`, `Right` or `Center`.
Background Image Scale	The magnification. from 0% up to 300%, to apply to the background image.

Canvas

Canvas Color	For charts with a grid and axes, the primary and optional secondary color of the background inside these axes. The default primary color is white. This property has both a sample color button that allows you to select a color and a field that lists the hexadecimal value(s) of the color(s) you have selected. If you want to use a gradient in the background, add a secondary color. Or delete the code for the primary color and add a different primary color.
Canvas Transparency	How opaque the primary or secondary background colors are inside the axes as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Canvas Background Ratio	Ratio, as a percentage, for the primary and secondary colors in the canvas background using a gradient.
Canvas Background Angle	Angle of the gradient of canvas background colors in degrees from 0-360. Default is 180 (horizontal gradient).
Canvas Border Color	The color of the canvas border.
Canvas Border Thickness	The thickness of the canvas border in points.
Canvas Border Transparency	How opaque the canvas border is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.

Show Canvas Background	Whether to show a background in the canvas. Defaults to `true`.
Show Canvas Base	Whether to show a base for the canvas in 3D charts. Defaults to `true`.
Canvas Base Color	The color for the canvas base in 3D charts.
Canvas Base Depth	The depth, in pixels, of the canvas base in 3D charts.
Canvas Background Depth	The depth of the canvas background in 3D charts.

Text Formatting

Base Font	Font family for all data labels within the chart. This is also the default font for captions and tick labels, unless the Caption/Tick Label Font property is set.
Base Font Size	Base font size in points, from 8-72, for all data labels within the chart. This is also the color for captions unless the Caption/Tick Label Font Size property is set.
Base Font Color	Color for all data labels within the chart. This is also the color for captions unless the Caption/Tick Label Color property is set.
Caption/Tick Label Font	Font family for captions and tick labels within the chart. If you omit this property, caption and tick labels default to the Base Font.
Caption/Tick Label Font Size	Font size in points, from 8-72, for captions and tick labels in the chart. If you omit this property, caption and tick labels default to the Base Font Size
Caption/Tick Label Color	Color for all captions and tick labels in the chart. If you omit this property, caption and tick labels default to the Base Font Color
Number of Decimals	The number of decimal places to use in all numeric labels in the chart.

Format Number Scale	Truncate numbers and use `K` (thousands) and `M` (millions) symbols to indicate proper values.
Format Number	Use thousand and decimal separators in numeric labels.
Force Decimals	Ensure all numeric labels have all decimal places, adding 0 where needed. If the number of decimals is 2, for example, a numeric value of 2.4 would display as 2.40 with this property set. This is false by default.
Decimal Separator	The symbol to use as the decimal point. Defaults to `dot` = a period.
Thousands Separator	The symbol to use as a separator for thousands and subsequent multiples. Defaults to `comma`.
Show Percent Values	Determines whether numeric values are shown as a percentage or a simple value. False by default.
X Axis Minimum Value	Explicitly sets the minimum value for the X axis.
X Axis Maximum Value	Explicitly sets the maximum value for the X axis.
Y Axis Minimum Value	Explicitly sets the minimum value for the Y axis.
Y Axis Maximum Value	Explicitly sets the maximum value for the Y axis.
Volume Y Axis Minimum Value	Explicitly sets the minimum value on the Y axis of the volume chart.
Volume Y Axis Maximum Value	Explicitly sets the maximum value on the Y axis of the volume chart.
Price Y Axis Minimum Value	Explicitly sets the minimum value on the Y axis of the Price chart.

Price Y Axis Maximum Value	Explicitly sets the maximum value on the Y axis of the Price chart.
Lower Limit Display	The text to display for the chart lower limit instead of the lower limit number.
Upper Limit Display	The text to display for the chart upper limit instead of the lower limit number.
Force Decimals Y Axis Values	Ensure all numeric labels for the Y axis have all decimal places, adding 0 where needed. If the number of decimals is 2, for example, a numeric value of 2.4 would display as 2.40 with this property set. This is false by default.
Y Axis Value Decimals	Sets the decimal precision for values in the Y axis.
Volume Force Decimals	Ensure all numeric labels have all decimal places, adding 0 where needed, in the Volume chart. If the number of decimals is 2, for example, a numeric value of 2.4 would display as 2.40 with this property set. This is false by default.
Volume Force Decimals Y Axis Values	Ensure all numeric labels in the Y axis have all decimal places, adding 0 where needed, in the Volume chart. If the number of decimals is 2, for example, a numeric value of 2.4 would display as 2.40 with this property set. This is false by default.
Volume Y Axis Value Decimals	Set the decimal precision for values in the Y axis of the Volume chart.
Volume Format Number	Use thousand and decimal separators in numeric labels in the Volume chart.
Volume Format Number Scale	Truncate numbers and use `K` (thousands) and `M` (millions) symbols to indicate proper values for numbers in the Volume chart.
Volume Number Decimals Places	The number of decimal places to use in all numeric labels in the Volume chart.

Volume Number Prefix	The character(s) to add as a prefix to numbers in the Volume chart.
Volume Number Suffix	The character(s) to add as a suffix to numbers in the Volume chart.

Layout

Chart Top Margin	Space at the top of the chart in pixels.
Chart Right Margin	Space on the right side of the chart in pixels.
Chart Bottom Margin	Space at the bottom of the chart in pixels.
Chart Left Margin	Space on the left side of the chart in pixels.
Canvas Top Margin	For charts with a grid and axes, the space in pixels at the top between the grid and any chart areas with captions or borders.
Canvas Right Margin	For charts with a grid and axes, the space in pixels to the right between the grid and any chart areas with captions or borders.
Canvas Bottom Margin	For charts with a grid and axes, the space in pixels at the bottom between the grid and any chart areas with captions or borders.
Canvas Left Margin	For charts with a grid and axes, the space in pixels to the left between the grid and any chart areas with captions or borders.
Logo URL	The URL to the logo image for the chart.
Logo Position	The position for your logo on the chart. This defaults to the top-left corner.
Logo Transparency	How opaque the logo is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.

Logo Scale	Change the scale (0-300) of an externally loaded logo at run-time
Logo Link	The URL to open when users click the logo in the chart. The format of the link determines whether the target URL opens in the same page or not: ■ *fully-qualified-url* = the target URL opens in the same brower page, replacing the view. ■ `N`–*fully-qualified-url* = the target URL opens in a new browser window. ■ `F`–*frame-name* –*fully-qualified-url* = the target URL opens in the named frame or iframe.
Show Shadow	Displays or hides (default) a shadow for the gauge in the chart.
LED Size	Sets the size of each LED bar in the gauge.
LED Gap	Sets the size of the gap between LED bars in the gauge.
Gauge Width	The horizontal space in pixels for a gauge.
Gauge Height	The vertical space in pixels for a gauge.

Labels

Show Labels	Determines whether text labels for the bars, lines, wedges or other graphical objects in this chart are displayed. True by default.
Show Values	Determines whether data values for the bars, lines, wedges or other graphical objects in this chart are displayed. False by default.
Show Tooltip	Whether to show a tool-tip with more information when the mouse hovers over data points. True by default
Label Display	Controls how text labels are displayed, in combination with **Slant Labels**:

	■ `Wrap` = the default format. This wraps long labels into multiple lines. Labels are displayed in a horizontal line. ■ `Rotate` = rotates labels to either a 45 or a 90 degree angle, based on the Slant Label property. ■ `Stagger` = alternates the line on which labels are displayed to provide more room for long labels. Labels are displayed in the number of lines defined by Stagger Lines.
Slant Labels	Determines the degree of text label rotation when **Label Display** is `Rotate`. The default is `true` which gives a 45 degree slant. If `false` labels are rotated 90 degrees. See also Rotate Values.
Label Step	How many long labels to skip on the X axis to allow labels to display fully
Stagger Lines	How many lines to stagger for long labels when Label Display is `stagger`. This defaults to 2.
Use Ellipses When Overflow	Whether to truncate long labels and display an ellipsis to indicate this. Defaults to true.
Rotate Values	Numeric values are displayed horizontally when `false` (the default). If `true`, numbers are rotated 90 degrees.
Rotate Labels	Whether to rotate long labels on the X axis. Defaults to true.
Value Position	Where to display the data value (`above`, `below` or `auto`) with respect to the data plot. Defaults to auto.
Show Y Axis Values	Whether the Y axis is divided into sections using division lines across the canvas. Defaults to true.
Show Limits	Show the limit values for the chart. Defaults to true.
Show Division Line Values	Show the values for divisional lines in the canvas.

Show Label Borders for Vertical Separator Lines	Show a border around labels for vertical separator lines.
Y Axis Values Step	The number of divisional line labels to skip to properly show long values.
Adjust Division	Whether divisional lines and limits can be adjusted to fit the chart data. Defaults to true.
Rotate Y Axis Name	Whether to rotate the Y axis label. Defaults to false.
Center Y Axis Name	Whether to vertically center the Y axis label. Defaults to true.
Y Axis Name Width	Sets a maximum width for the Y axis label.
Draw Vertical Joins	Whether to draw vertical lines for joins on the canvas. Defaults to true.
Show Zero Plane Value	Whether to show the value of the zero plane. Defaults to true.
Show Tooltip for Wrapped Labels	Whether to show tooltips for labels that are shortened. Defaults to true.
Show Tooltip	Whether to show tooltips with more information for data points or plots in the canvas.
Tooltip Background Color	The color to use as a background for tooltips (see Show Tooltips for more information). Defaults to white.
Tooltip Border Color	The color to use as a border around tooltips (see Show Tooltips for more information). Defaults to white.
Tooltip Separator Character	The character, such as a comma, to separate the name and value displayed in the tooltip.
Series Name in Tooltip	Determines whether the names for the series, columns or fields that are plotted in the chart appear in tooltips. Defaults to no.

Show Tooltip Shadow	Determines whether tooltips should appear with a drop shadow. Defaults to no.
Show Legend	Determines whether to include a legend relating colors used in plots to the series, columns or fields that are represented by that color. Defaults to yes.
Legend Caption	Enter a caption, if needed, for the legend. This is empty by default.
Legend Background Color	The color to use as a background for the legend. Defaults to white. This property has a sample color button that allows you to select a color.
Legend Position	■ `Bottom` = centered at the bottom of the canvas area. This is the default. ■ `Right` = vertically centered to the right of the canvas area.
Legend Allow Drag	Determines whether users can drag the legend to other positions in the chart. This defaults to no.
Legend Marker Circle	Determines whether the legend uses square markers to show colors or circles. Default to no, which means square color markers are used.
Reverse Legend	Determines whether the order of colors is reversed or not. Defaults to no, indicating that colors are shown in the same order in which you added series, columns or fields to the chart.
Legend Background Transparency	How opaque the legend background is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Legend Border Color	The color to use in the border around the legend. Defaults to white. This property uses a sample color button that allows you to select a color.
Legend Border Thickness	The thickness, in pixels of the border. This is empty by default, indicating no border.

Legend Border Transparency	How opaque the legend border is as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Legend Shadow	Whether the legend should have a drop shadow. This defaults to yes.
Legend Scroll Background Color	The color to use as the background for the scroll bar . Defaults to white. This property uses a sample color button that allows you to select a color.
Legend Scroll Bar Color	The color to use in the bar of the scroll bar. Defaults to white. This property uses a sample color button that allows you to select a color.
Legend Scroll Button Color	The color to use in up/down buttons of the scroll bar. Defaults to white. This property uses a sample color button that allows you to select a color.
Enable Smart Labels	Whether to use (`true` the default) smart lines in funnel charts.
Caption Padding	Space in pixels between the caption and/or sub-caption and the top of Pyramid charts.
Pyramid Y Scale	Sets the three dimensional perspective of the pyramid, in degrees from 0-40.
Show Tick Marks	Displays ticks within the scale of the chart to help relate the size or placement of chart objects to a scale.
Show Tick on Right	Whether to show tick marks on the right (true) or left in gauges. Defaults to true
Show Tick Labels	Displays tick values. True by default.
Tick Value Distance	Distance between tick values and tick marks.

Advanced

Palette Colors	Allows you to pick multiple colors to use as fill colors for plots (columns, bars, wedges, lines, areas and other graphic shapes) in some charts.

	To add a color: 1. Click the color swatch. 2. Find a color in the window that opens and close the window when you have the right color. 3. Click **Add Color**. This adds a swatch to the list along with the hexadecimal value for this color. Colors get added in order and correspond to specific series of data. For pie or doughnut charts, the list of colors simply repeats through the wedges. Click a color in the list to delete it.
Palette	Pre-defined color palettes for chart backgrounds and labels.
Color Range Palette	For dial gauges, determines the list of colors used for each range within the dial.
Animation	Determines whether the chart rendering is animated (`true` by default) or not. Nonanimated rendering may pause until the entire chart can be displayed.
Smart Line Color	For pie and doughnut charts, the color of lines connecting labels to a wedge.
Smart Line Thickness	For pie and doughnut charts, the thickness in pixels of lines connecting labels to a wedge.
Smart Line Transparency	For pie and doughnut charts, the opacity, from 0-100, as a percentage of lines connecting labels to a wedge. Zero is fully transparent and 100 is fully opaque. If not set, this is fully opaque.
Smart Line Slanted	For pie and doughnut charts, determines how the lines connecting labels to wedges are bent. `Slanted` is the default which allows different angles. `Slanted` uses 90 degree bends only.
Round Radius	Sets the radius to use for rounded corners.
Is Sliced	Determines whether funnel slices are sliced-out or not (default).

Is Hollow	Determines whether funnel slices are hollow or filled (default).
Stream Lined Data	Whether to present the data as streamlined data or as section data. Section by default.
Use Same Slant Angle	Assigns identical slant angles to all slices.
Show Plot Border	Whether to show a border around two-dimensional funnel charts.
Plot Border Thickness	Border thickness in pixels for two-dimensional funnel charts.
High Color	The color to use for the high value column in Sparkline Column charts.
Low Color	The color to use for the low value column in Sparkline Column charts.
Win Color	The color to use for win columns in Win/Loss Sparkline charts.
Loss Color	The color to use for loss columns in Win/Loss Sparkline charts.
Radar Radius	Sets a best-fit radius for radar chart plots. If this is not set, the best fit is automatically calculated.
Draw Anchors	Show anchors for data points in a radar chart plot.
Anchor Sides	Sets the number of sides, from 3 up to 20, to use in anchors.
Anchor Radius	Sets the radius, in pixels, for anchors.
Anchor Border Color	The color to use in the border of anchors.
Anchor Border Thickness	The thickness, in pixels, of anchor borders.

Anchor Background Color	The background (or fill) color or anchors.
Anchor Transparency	How opaque anchors are as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Anchor Background Transparency	How opaque anchor backgrounds are as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Interactive Legend	Whether users can click in the legend to visually hide specific series. This is true by default.
Minimize Wrapping in Legend	Whether to minimize text wrapping in the legend. This is true by default.
Legend Column Numbers	Sets the number of columns to show with labels in the legend.
Data Line Color	The color to use in plot lines for data.
Data Line Thickness	The thickness, in pixels, of plot lines for data.
Data Line Transparency	How opaque plot lines for data are as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Data Line Dashed	Whether plot lines for data should be dashed. Defaults to false.
Data Line Dash Length	The length, in pixels of dashes in a plot line when dashing is enabled.
Data Line Dash Gap	The distance, in pixels, between dashes in a plot line when dashing is enabled.
Show Radar Border	Whether to show the outer border in radar charts.
Radar Border Color	The color to use for the outer border in radar charts.
Radar Border Thickness	The thickness, in pixels, for the outer border in radar charts.

Radar Border Transparency	How opaque the outer border for a radar chart should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Radar Fill Color	The fill color for radar plots.
Radar Fill Transparency	How opaque the fill for radar plots should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Radar Spike Color	The color to use for vertices (lines that radiate from the center) in radar plots.
Radar Spike Thickness	The thickness, in pixels, of vertices (lines that radiate from the center) in radar plots.
Radar Spike Transparency	How opaque vertices (lines that radiate from the center) in radar plots should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Reversal Value	The specific value that determines when a horizontal shift should occur to plot the next point.
Reversal Percentage	The percentage of change that determines when a horizontal shift should occur to plot the next point.
Maximum Horizontal Shift Percent	Sets the maximum horizontal shift as a percentage.
Rally Color	The color to use for plot lines that indicate a rally.
Rally Thickness	The thickness, in pixels, for rally lines in the chart.
Rally Transparency	How opaque rally lines should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Rally Dashed	Whether rally lines should be dashed.
Rally Dash Length	The length, in pixels, for rally line dashes when dashing is enabled.

Rally Dash Gap	The distance, in pixels, between dashes in rally lines when dashing is enabled.
Decline Color	The color to use for plot lines that indicate a decline.
Decline Thickness	The thickness, in pixels, for decline lines in the chart.
Decline Transparency	How opaque decline lines should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Decline Dashed	Whether decline lines should be dashed.
Decline Dash Length	The length, in pixels, for decline line dashes when dashing is enabled.
Decline Dash Gap	The distance, in pixels, between dashes in decline lines when dashing is enabled.
Plot Price As	What type of plot to use for the price chart: ■ `Default` ■ `Candlestick` ■ `Line` ■ `Bar`
Plot Closing Price	Whether the closing price is included in the price chart.
Show Volume Chart	Whether to show the volume chart.
Volume Height Percent	The percentage of the overall chart, from 20% to 80%, that the volume chart should occupy. See also Plot Space Percent.
Plot Space Percent	The percentage of the overall chart that the price chart should occupy. See also Volume Height Percent.
Bear Fill Color	The fill color for candles or bars with a bear trend.
Bear Border Color	The border color for candles, wicks (lines) or bars with a bear trend.

Bull Fill Color	The fill color for candles or bars with a bull trend.
Bull Border Color	The border color for candles, wicks (lines) or bars with a bull trend.
Rollover Band Color	The color of the rollover band.
Rollover Band Transparency	How opaque the rollover band should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Show Volume Chart Plot Borders	Whether to show plot borders in the Volume chart.
Volume Chart Plot Border Color	The border color to use for plots in the Volume chart.
Volume Chart Plot Border Thickness	The border thickness, in pixels, for plots in the Volume chart.
Volume Chart Plot Border Transparency	How opaque the plot borders in the Volume chart should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Plot Line Transparency	How opaque the plot lines should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Plot Line Thickness	The thickness, in pixels, for plot lines.
Plot Line Dash Length	The length, in pixels, for dashes in plot lines.
Plot Line Dash Gap	The distance, in pixels, between dashes in plot lines.
Maximum Column Width	Sets the maximum width for columns.
Use 3D Lighting	Turns on advanced gradients and shadows in 3D charts.
Primary Y Axis Name Width	Sets a maximum width for the primary Y axis label.

Secondary Y Axis Name Width	Sets a maximum width for the secondary Y axis label.
Place Values Inside	Whether data values are placed inside columns.
Show Secondary Limits	Whether to show limits for the secondary Y axis.
Show Divisional Line Secondary Value	Whether to show the secondary axis value for divisional lines in the canvas.
Show Cumulative Line	Whether to plot lines as cumulative values for the series.
Show Line Values	Whether to display data point values on lines.
Plot Border Color	The color to use for borders for plots in the chart.
Plot Border Transparency	How opaque the plot borders should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Plot Border Dashed	Whether plot borders should be dashed.
Plot Border Dash Length	The length of dashes, in pixels, for the plot border when dashing is enabled.
Plot Border Dash Gap	The distance, in pixels, between dashes in plot borders when dashing is enabled.
Plot Fill Angle	Sets the angle (0 to 360) for plot fills that use gradients.
Plot Fill Transparency	How opaque the plot fill should be as a percentage (0-100%). Zero is fully transparent and 100 is fully opaque.
Plot Gradient Color	A global color to use in gradient fills for plots in the chart.
Plot Fill Ratio	Sets the fill ratio for gradients in plots for the chart.

Use Round Edges	Sets rounded corners for columns and fills columns with a glass effect gradient.
Show Sum	Show the sum of all segments in a stacked column layout.
Use Percent Distribution	Show data values as percentages.
Show X Axis Percent Values	Show values as percentages along the X axis.
Overlap Columns	Overlap columns in 3D layouts for a richer look.
Gantt Pane Duration Unit	The time unit that is used in a Gantt chart timeline. This defaults to Month.
Gantt Pane Duration	The number of time units to include in a Gantt chart timeline.
Slack Fill Color	If you include percentage of completion statistics for a Gantt chart, this is the color used to fill the bars for the uncompleted portions of tasks.
Show Slack as Fill	Determines whether uncompleted portions of tasks are shown with fill color or not. This defaults to true. This is only applicable if you include percentage of completion statistics for the Gantt chart.
Show Percent Label	Determines whether a label with the percentage of completion is shown for tasks in a Gantt chart. Defaults to No. This is only applicable if you include percentage of completion statistics for the Gantt chart.
Show Task Start Date	Determines whether a label with the start date is shown for tasks in a Gantt chart. Defaults to No.
Show Task End Date	Determines whether a label with the end date is shown for tasks in a Gantt chart. Defaults to No.

More Properties

Attributes	Additional chart attributes. Here you can add any attribute to the `< chart more attributes …. / >` tag to extend the properties of the chart section, e. g. apply new chart themes.
Attribute name	Name of the new attribute, e. g. theme.
Attribute value	Value of the new attribute, e. g. carbon.
Add Attribute	Add the attribute (name/value) to the chart properties.
Delete Selected Attributes	Delete the selected attribute from the chart properties.
More Attributes	List of additional chart attributes. For information on Fusion Chart attributes, see documentation for: ■ FusionCharts XT 3.3.1 ■ FusionWidgets XT 3.3.1

Creating Pluggable Views or Libraries

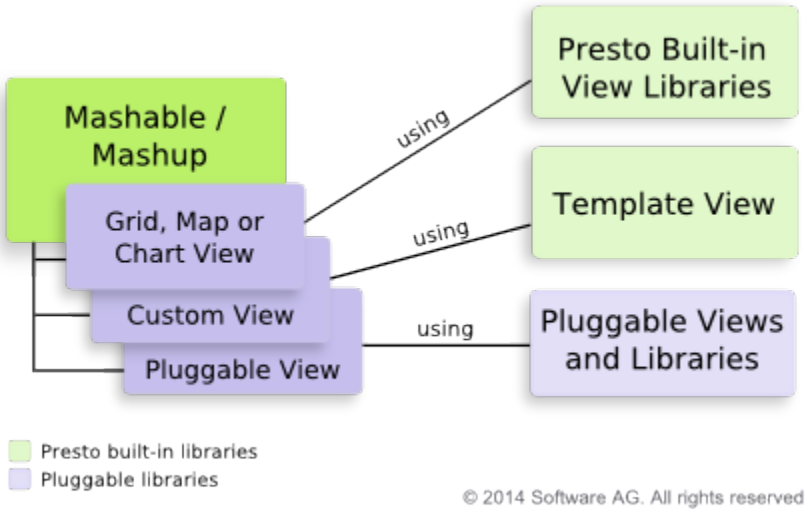
For more information on managing pluggable views and libraries, see "Managing Updates and Library Versions" on page 1085 .

Views and Libraries in MashZone NextGen

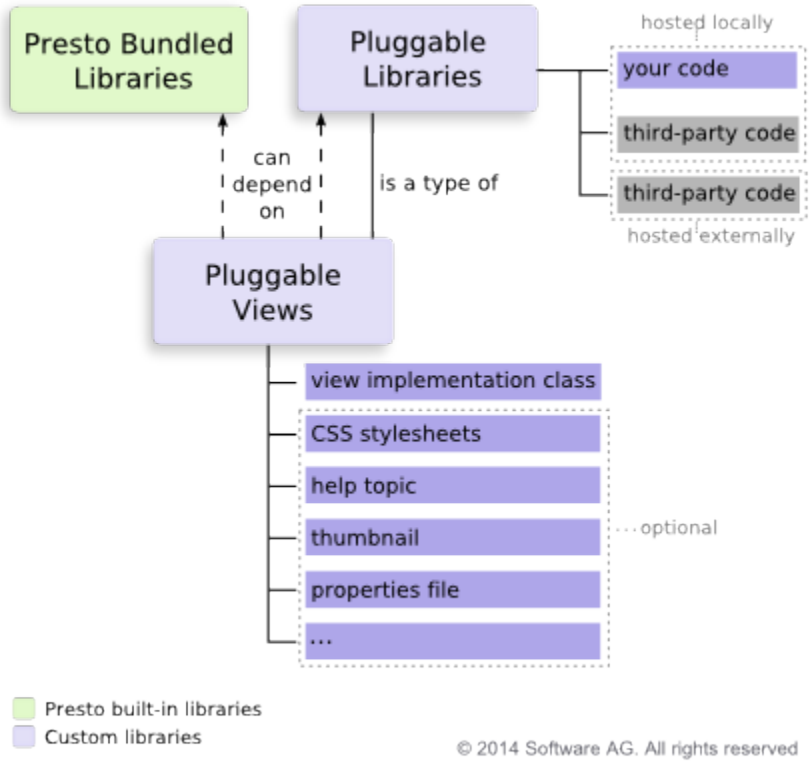
MashZone NextGen contains a set of *libraries* that produce the built-in MashZone NextGen views with many of the most common presentation forms, such as grids, maps or charts, for mashable or mashup results. These different presentations are listed as built-in views in the View Gallery of the MashZone NextGenView Maker wizard so users can easily create views for specific mashables and mashups.

In addition to the built-in views, users can also create custom views for a mashable or mashup using the built-in "Template View" on page 1006. These custom views are specific to that mashable or mashup and are *not* reusable.

MashZone NextGen developers can also extend the list of available, reusable views in the View Gallery by creating *pluggable views* and adding them to MashZone NextGen.



Pluggable views contain resources, the JavaScript, CSS, HTML or image files, needed to render data in that specific format.



Pluggable views must implement theMashZone NextGenViews API. They can have additional resources and configuration to allow them plug into the MashZone NextGen View Gallery and allow you to manage them in MashZone NextGen.

They can also use other libraries including libraries bundled in MashZone NextGen or *pluggable libraries* that MashZone NextGen developers add to MashZone NextGen.

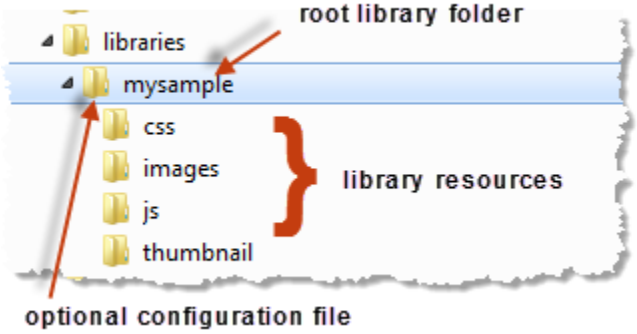
Pluggable libraries may contain any of the common types of web resources (JavaScript, CSS, etc.). They can be written specifically for your organization or they can be provided by third parties.

Important: MashZone NextGen includes certain "*third party software*" which JackBe licenses from third parties. Pursuant to the JackBe EULA for MashZone NextGen, all MashZone NextGen users are bound by the license terms and conditions of any third party software licenses. You may review these Third Party Licenses at http://documentation.softwareag.com/legal/.

Software AG does not and cannot authorize any use of third party software that is not permitted by these third party software licenses. You may, however, be able to obtain licenses directly from third parties. In addition, any other software that you add and/or use in connection with MashZone NextGen is subject to its own licensing requirements and may void the terms of the EULA for MashZone NextGen.

A pluggable view or library has a folder structure to hold its local resources, as shown below:

Pluggable View or Pluggable Library Folders



The name of the root folder is also the ID for that view or library. Pluggable views and libraries may contain local resources or they may point to resources that are hosted externally.

Implement and Add a Pluggable Library or View

To create and add a pluggable library or pluggable view

1. Create the folders needed for the pluggable library or view. See "Figure 5" on page 1047. The root folder for the pluggable view or library must match the view or library ID.
2. Write or download the code needed for this view or library:
   - For pluggable views, you must Implement a Pluggable View Class using the MashZone NextGen View API and add this to the library's /js folder.

- For pluggable libraries that use resources hosted externally, skip this step.
- For all other pluggable libraries, you may download resources for third-party libraries and add them to the library's folders. Or write the necessary JavaScript code and add it to the /js folder.

3. Optionally, add CSS Styles, Help and Thumbnails for Pluggable Views or any other resources needed for the pluggable library or pluggable view.
4. Optionally, define additional configuration that MashZone NextGen may need to import and manage the view or library. Configuration is defined in either:
   - A *library properties* file that you save in the root library folder. See "Import Pluggable Library/View Examples" on page 1076 for more information and examples.
   - An Apache Ant™ build file to import pluggable views and libraries if you have Ant in your development environment. See "Ant Tasks and Sample Build File to Import Pluggable Views or Libraries" on page 1074 for more information and examples.

   See "Configuration Properties for Pluggable Views or Libraries" on page 1067 for information on configuration properties.

   Additional configuration is *required* to register a pluggable library as a pluggable view or to register a pluggable library with external resources. You may also use library configuration to:
   - Define basic metadata such as a description and version or identifying a pluggable library as a pluggable view.
   - Identify images and help to use with a pluggable view library.
   - Declare dependencies for this library or view on libraries bundled in MashZone NextGen or other custom or third-party libraries added to MashZone NextGen.
   - Define the external resources for pluggable libraries that have no local resources.
5. Upload the pluggable view or pluggable library using either the MashZone NextGen `importLib` command or using Ant with a build file. See "Import Pluggable Views and Libraries" on page 1073 for examples and instructions.

Implement a Pluggable View Class

This topic presents the basic requirements of implementing a pluggable view using A Hello World Pluggable View Class as an example. For other common requirements and examples, see "Pluggable View Classes: Using Mashable/Mashup Data" on page 1054 and "Pluggable View Classes: Triggering and Handling Events" on page 1059.

A Hello World Pluggable View Class

The library for each pluggable view must contain a class that implements or uses theMashZone NextGenViews API for:

- A constructor. See "Pluggable View Constructor" on page 1049 for an example.
- A method to render the view. See "Render the Pluggable View" on page 1050 for an example.
- Optionally, a method to update the view.
- A method to render a configuration input fields used in one step in the MashZone NextGenView Maker wizard. These inputs collects configuration information when users add or edit views for mashables or mashups based on this pluggable view. See "Support Configuration for a Pluggable View" on page 1051 for an example.
- A method to collect user entries for view configuration.
- An optional method to validate user configuration entries.
- A method to Register this Pluggable View in the MashZone NextGen View framework.

For the complete hello world sample, see "A Hello World Pluggable View Class" on page 1048.

Pluggable View Constructor

In this example, we start the class for this pluggable view as a self-invoking function to implement the view in a closure and associate the `$` method with jQuery for this view. (This is not a requirement for pluggable views, just one possible design choice.)

```
(function($){
  //set id and namespace
  //constructor
  //register view
}(jQuery));
```

The class has three parts: setting basic properties (the ID and namespace), defining the constructor and view implementation, and registering the view.

Initial setup defines the ID for this view library which generally should match the root folder name for the library. It also creates a namespace for this pluggable view using the `Presto.namespace` method.

Tip: Defining pluggable view classes in your own namespace prevents potential conflicts with other JavaScript libraries.

```
(function($){
  var id="hello-world";
  Presto.namespace("Sample.view");
  //constructor
  //register view
}(jQuery));
```

Next we add the constructor as a method defined in the namespace for this class. The constructor's signature is *`view-class-name`*`(`*`selector`*`, `*`dataTable`*`, [`*`config`*`])`:

```
(function($){
  var id="hello-world";
  Presto.namespace("Sample.view");
  Sample.view.HelloWorld = function(selector, dataTable, config) {
  };
```

```
  //register view
}(jQuery));
```

The `selector` parameter identifies the *view container* - the node in the containing app or page where this view will be rendered. This can be a CSS selector pointing to the node or it can be a DOM node.

The actual data and data model that the view will use is passed in the `dataTable` parameter. This is a MashZone NextGen DataTable object. See theMashZone NextGenDataTable API for more information.

The last parameter, `config`, is an optional object with configuration properties for this view with this mashable or mashup. This is the configuration that users set when they add views or edit them in MashZone NextGen Hub.

Lastly, we initialize some variables and properties and make sure the selector is a jQuery node for easy access later:

```
(function($){
  var id="hello-world";
  Presto.namespace("Sample.view");
  Sample.view.HelloWorld = function(selector, dataTable, config) {
     config = config || {};
     var self = this, el = $(selector);
     self.config = config;
    //draw method
    //update method
    //showConfig method
    //getConfig method
    //validate method
  };
  //register view
}(jQuery));
```

In this example constructor, we will also implement methods from the View API to render the view and to allow users to configure the view. See "Render the Pluggable View" on page 1050 and "Support Configuration for a Pluggable View" on page 1051.

Render the Pluggable View

To render the view, pluggable view libraries:

- Must implement `draw(dataTable, [config])` to render the view initially.
- Can optionally implement `update(dataTable, [config])` to render the view when there are updates to the `dataTable` (the data or model) or other events.

For this example, the view is a single HTML element with one dynamic value for the name to add to the greeting. The value to render is defined as a property in the `config` parameter object, from user configuration for the view, rather than from mashable or mashup results that are normally passed in the `dataTable` parameter.

```
(function($){
  var id="hello-world";
  Presto.namespace("Sample.view");
  Sample.view.HelloWorld = function(selector, dataTable, config) {
    config = config || {};
    var self = this, el = $(selector);
    self.config = config;
    self.draw = function(dataTable, config){
```

```
      el.html('<div
><p>Hello ' + config.greeting + '</p></div>');
    };
    self.update = function(dataTable, options){
      el.html('<div
><p>Hello ' + config.greeting + '</p></div>');
    };
    //showConfig method
    //getConfig method
    //validate method
  };
  //register view
}(jQuery));
```

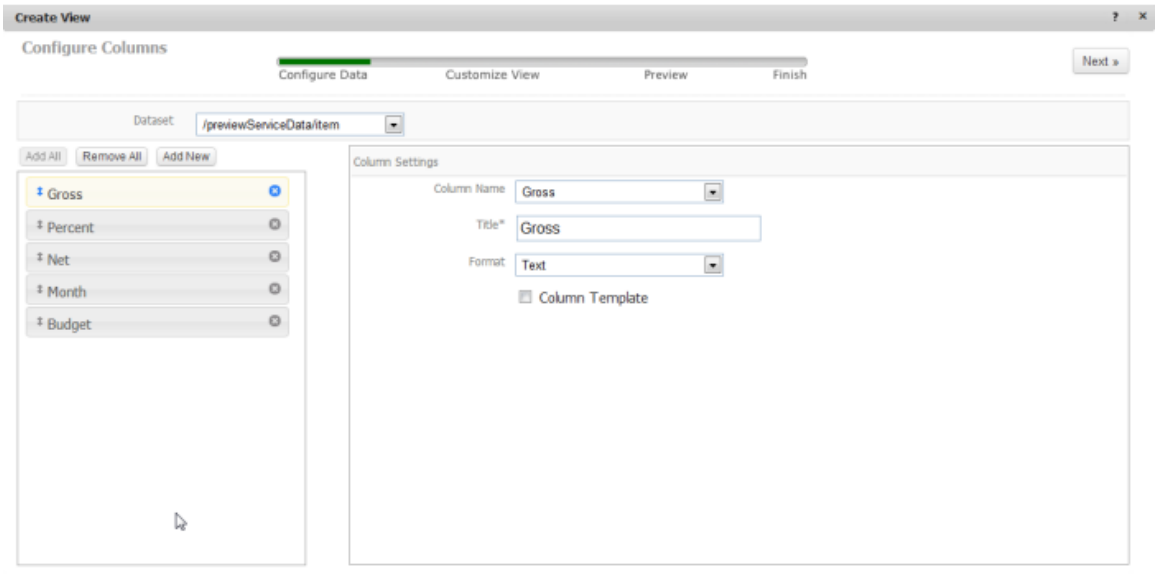
Support Configuration for a Pluggable View

In almost all cases, pluggable views will have configuration information that users must set to enable the view to render results from specific mashables or mashups. Configuration may also include specific visual aspects to optimize the view.

To complete view configuration in the MashZone NextGenView Maker wizard

1. Data configuration is provided automatically by the View Maker wizard to let users determine the dataset to use in the view, delete columns of no interest for this view and handle other common configuration for specific columns.

   See the "Pluggable View Classes: Using Mashable/Mashup Data" on page 1054 example for more information on how this view configuration is used in pluggable views.



2. View configuration specific to this pluggable view is provided by the pluggable view implementation class using the `showConfig(`*`dataTable`*`, `*`selector`*`, `*`config`*`)` to render input fields and `getConfig()` methods to retrieve user entries. You can also optionally implement the `validate()` method to validate user configuration.

For this Hello World pluggable view, configuration inputs look like this:

3. Preview is provided automatically by the View Maker wizard with a rendering of the view based on the current data and configuration.
4. Finish is provided automatically by the View Maker wizard to allow users to name and save the view.

The `showConfig` method defines the input fields that users complete in the View Configuration step of the View Maker wizard and also makes sure that any existing configuration for properties fills these fields when users edit existing views.

For Hello World, this is a single input field where users enter a name for the greeting:

```
...
  Sample.view.HelloWorld = function(selector, dataTable, config) {
    ...
    self.showConfig = function(dataTable, selector, config){
      self.form = $(selector).html('<div>Greeting: <input type="text" name="greeting" /></div>');
      self.form.find('input[name=greeting]').val(config.greeting || self.config.greeting || '');
     };
    //getConfig method
    //validate method
  };
  //register view
}(jQuery));
```

The `selector` parameter is either a CSS selector or the DOM node where this form is rendered in the View Maker wizard. The `dataTable` parameter is both the data and the data model for the current mashable or mashup.

The final parameter, `config`, is the configuration object that is updated with user configuration choices for this view.

Important: Since users can edit existing views with existing configuration as well as users moving back through previous steps in the wizard when they add views, the `showConfig` method *must* support the case where values may already exist in the `config` parameter for each configuration field.

The View Maker wizard uses the `getConfig` method to populate the configuration object for the view with user entries from these fields when users click the **Next** button to move to Preview.

```
...
  Sample.view.HelloWorld = function(selector, dataTable, config) {
    ...
    self.showConfig = function(dataTable, selector, config){
      self.form = $(selector).html('<div>Greeting: <input type="text" name="greeting" /></div>');
      self.form.find('input[name=greeting]').val(config.greeting || self.config.greeting || '');
```

```
     };
    self.getConfig = function(){
      if(self.form){
          self.config.greeting = self.form.find('input[name=greeting]').val();
      }
      return self.config;
    };
    //validate method
  };
  //register view
}(jQuery));
```

If the `validate` method is also defined, the View Maker wizard validates the user entries before calling `getConfig`.

```
...
  Sample.view.HelloWorld = function(selector, dataTable, config) {
    ...
    self.showConfig = function(dataTable, selector, config){
      self.form = $(selector).html('<div>Greeting: <input type="text" name="greeting" /></div>');
      self.form.find('input[name=greeting]').val(config.greeting || self.config.greeting || '');
     };
    self.getConfig = function(){
      if(self.form){
          self.config.greeting = self.form.find('input[name=greeting]').val();
      }
      return self.config;
    };
    self.validate = function(){
      var config = self.getConfig();
      return config.greeting ? true : false;
    };
  };
  //register view
}(jQuery));
```

Register this Pluggable View

Once the constructor and view class implementation is done, the last step is to use the `Presto.view.register(`*`viewConfig`*`)` method in the MashZone NextGen View API to make the pluggable view visible in the MashZone NextGen View framework:

```
(function($){
  var id="hello-world";
  Presto.namespace("Sample.view");
  Sample.view.HelloWorld = function(selector, dataTable, options) {
    ...
  };
  Presto.view.register({
    cls: Sample.view.HelloWorld,
    lib: id
  });
}(jQuery));
```

The configuration object for this method contains two properties to identify the view library ID and the name of the implementation class.

Complete Sample.view.HelloWorld Pluggable View Class

For more information on this class see "Pluggable View Constructor" on page 1049, "Render the Pluggable View" on page 1050, "Support Configuration for a Pluggable View" on page 1051 and "Register this Pluggable View" on page 1053.

```
(function($){
  var id="hello-world";
  Presto.namespace("Sample.view");
  Sample.view.HelloWorld = function(selector, dataTable, config) {
    config = config || {};
    var self = this, el = $(selector);
    self.config = config;
    self.draw = function(dataTable, config){
      el.html('<div
><p>Hello ' + config.greeting + '</p></div>');
    };
    self.update = function(dataTable, options){
      el.html('<div
><p>Hello ' + config.greeting + '</p></div>');
    };
    self.showConfig = function(dataTable, selector, config){
      self.form = $(selector).html('<div>Greeting: <input type="text" name="greeting" /></div>');
      self.form.find('input[name=greeting]').val(config.greeting || self.config.greeting || '');
     };
    self.getConfig = function(){
      if(self.form){
          self.config.greeting = self.form.find('input[name=greeting]').val();
      }
      return self.config;
    };
    self.validate = function(){
      var config = self.getConfig();
      return config.greeting ? true : false;
    };
  };
  Presto.view.register({
    cls: Sample.view.HelloWorld,
    lib: id
  });
}(jQuery));
```

Pluggable View Classes: Using Mashable/Mashup Data

To render a pluggable view in most cases, the pluggable view class must map columns from response data for the view's mashable or mashup to particular visual aspects of the view. To render a pie chart, for example, a view must know which field has the label for each slice and which field has the value to use to calculate the arc for the pie slice.

Note: This topic discusses specific aspects of how to implement pluggable view classes. For the basic implementation techniques, see "Implement a Pluggable View Class" on page 1048.

MashZone NextGen views use a MashZone NextGen DataTable instance as the client-side data model of a tabular dataset contained in the mashable or mashup response. The MashZone NextGenView Maker wizard also uses Data Table information to let users

configure which dataset columns serve specific purposes for the view. Pluggable views can make use of both during view configuration and view rendering.

This example allows users who create views based on this pluggable view to select one column as configuration for the view in the MashZone NextGenView Maker wizard. It then uses this selected column configuration, plus any formatting configuration that users may have set for dataset columns and the Data Table with response data to access and format the correct data to render:

```
(function($){
  var id = "show-column";
  Presto.namespace('Sample.view');
  Sample.view.ShowColumn = function(selector, dataTable, config){
    config = config || {};
    var self = this, el = $(selector);
    self.config = config;
    self.draw = function(dataTable, config){
var column = config.selectedColumn,
rows = Presto.view.getRows(dataTable, config.columns, {"flatten":"true"});
if(rows && rows.length && column){
el.empty();
el.append('<div
>' + column + '</div>');
rows.each(function(row, i){
el.append('<div
>' + row[column] + '</div>');
});
      }
    };
    self.showConfig = function(dataTable, selector, config){
      config = config || self.config;
      var form = $(selector);
      self.form = form;
form.html('<div><div>Select Column:</div><div><select></select></div></div>');
config.columns.each(function(col){
var name = col.name,
selected = name == config.selectedColumn ? 'selected="selected"' : '';
form.find('select').append('<option value="' + name + '" ' + selected + ' >' + name + '</option>'
});
    };
    self.getConfig = function(){
      if(self.form){
self.config.selectedColumn = self.form.find('select').val();
      }
      return self.config;
    };
    self.validate = function(){
      var config = self.getConfig();
return config.selectedColumn ? true : false;
    };
 };
|
Presto.view.register({
  cls: Sample.view.ShowColumn,
  lib: id
});
}(jQuery));
```

Selecting Columns for View Configuration

The `showConfig` method is called in the second step of the MashZone NextGenView Maker wizard. The Data Table with the mashable/mashup response is passed as well as a configuration object for the view and a selector for the node in this page where the configuration form will be rendered.

```
self.showConfig = function(dataTable, selector, config){
  config = config || self.config;
  var form = $(selector);
  self.form = form;
  form.html('<div><div>Select Column:</div><div><select></select></div></div>');
  config.columns.each(function(col){
    var name = col.name,
    selected = name == config.selectedColumn ? 'selected="selected"' : '';
    form.find('select').append('<option value="' + name + '" ' + selected + ' >' + name + '</
  });
};
```

The `config` parameter may contain:

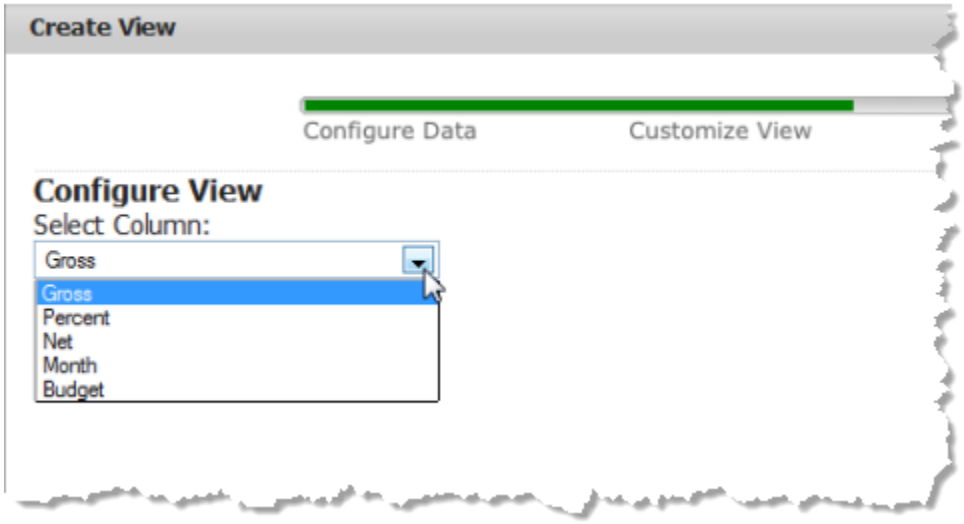
- Properties for the view itself, such as the name, and the underlying view library.
- Properties for the fields in this configuration form, if the user is editing an existing view or has simply moved back to a previous step in the wizard.
- A `columns` property with an array of objects containing configuration for each column that the user has chosen to include in this view in the previous step (Data Configuration). Each configuration object for a column can contain these properties:

name	The name of this column. If the data for this row or cell is complex, this may be a path separated by / marks.
header	The label to display for this column. This defaults to the column name, but users can change this to provide a more useful display label.
data-type	One of several formats for the data in this column. ■ `date` = a date and the format to show it in (either mm/dd/yyyy or dd/mm/yyyy) ■ `html` = the data also contains HTML markup tags which should be rendered ■ `image` = a URL to an image to show in an <img> tag. ■ `link` = a URL to make into a link with an <a> tag. ■ `longstring` = long text content will be truncated and ellipsis added ■ `number` = numeric data and the formats to apply

	■ `string` = text content. This is the default
template	An optional template of how to construct the data for this column.

Note: The MashZone NextGenView Maker wizard shows users options to set alternate formats and templates for columns in the Data Configuration step. You can easily apply this configuration when the view is rendered in the `draw` or `update` methods. See "Selecting Columns for View Configuration" on page 1056 for an example.

This `columns` array is used to build the options for a <select> field in this form so that users can select one column, such as this example:



The `getConfig` method is called when users click **Next** to move to the Preview step in the View Maker wizard. This method retrieves the name of the selected column and saves it in the `config` object for the view.

```
      if(self.form){
        self.config.selectedColumn = self.form.find('select').val();
      }
```

Formatting and Rendering Response Data

When the draw method renders the view, both the configuration object for that view and the DataTable with the mashable/mashup response are passed as parameters. This example retrieves the name of the column that was selected for this view from `config.selectedColumn`.

```
    self.draw = function(dataTable, config){
var column = config.selectedColumn,
      rows = Presto.view.getRows(dataTable,config.columns,{"flatten":"true"});
      if(rows && rows.length && column){
        el.empty();
        el.append('<div
```

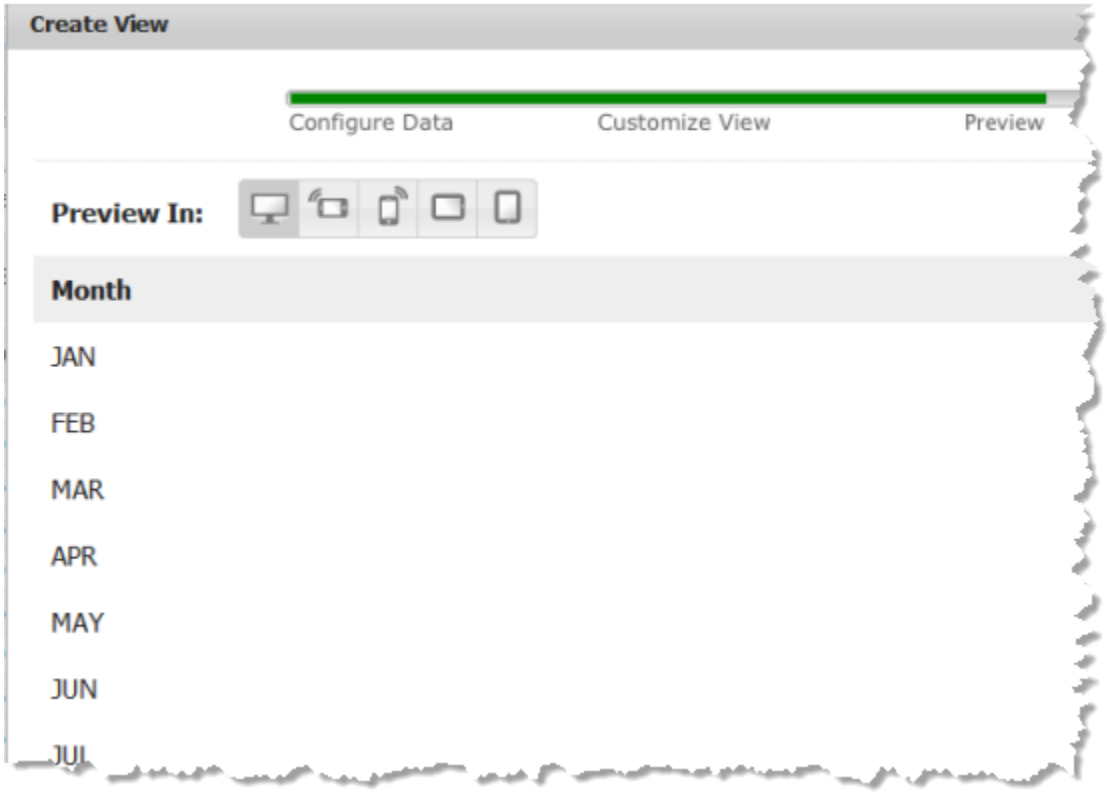
```
>' + column + '</div>');
        rows.each(function(row, i){
          el.append('<div
>' + row[column] + '</div>');
        });
      }
    };
```

It retrieves the data to render and applies any configuration for the columns of this view to the data using the `Presto.view.getRows` method. Users define column configuration options in the Data Configuration step of the MashZone NextGenView Maker wizard. This configuration is stored in the `columns` property of view configuration.

Note: You can also obtain the raw data from the `rows` field in the DataTable.

The formatted data that is returned by `Presto.view.getRows` is an array of objects, each object representing one row of data for the dataset that the user selected for this view. (The dataset is selected in the Data Configuration step of the MashZone NextGenView Maker wizard.)

Finally, it steps through each row and appends the appropriate HTML to render the name of the column and the data in each row for that column. The preview of this view for a mashable with a column named `Month` would look like this:



Pluggable View Classes: Triggering and Handling Events

Pluggable views may be used in basic apps with other views. Apps, in turn, may be used by themselves or included in workspace apps with other apps. Views may have events that occur to view elements which they must handle, but may also be of interest to other views or apps in different contexts. In turn, views may also be interested in events from other views or apps.

Note: This topic discusses specific aspects of how to implement pluggable view classes. For the basic implementation techniques, see "Implement a Pluggable View Class" on page 1048.

To support inter-view communication, pluggable views can use their DataTable. The DataTable tracks row selection state for the results in the DataTable across all views that use that DataTable and also supports both firing and listening for row selection events.

Apps may also leverage view events and communication to provide inter-app communication through the MashZone NextGen App Framework. The MashZone NextGen App Framework supports publishing and subscribing to events (called *topics*) across apps.

Thus, pluggable views can choose to support two general categories of events:

- *View-specific events* typically based on common browser events such as click for elements in the view.

  To enable view-specific events, you must Register and Trigger View-Specific Events in Pluggable Views.

- *Row selection events* for the view's DataTable. See "Support DataTable Row Selection Events in Pluggable Views" on page 1061 for details.

  Supporting DataTable events allows pluggable views to participate in *row selection synchronization* in basic apps. Virtually all MashZone NextGen built-in views participate in this synchronization where row selection in one view is automatically reflected in all other views, as appropriate, within a basic app.

  DataTable row selection events are also highly useful for users to wire interactions in workspaces because these events include the data of the entire row, not just those columns included in the view.

See "Example: Complete D3 Pie with View-Specific and DataTable Events" on page 1063 for the fully completed example of both types of events.

Register and Trigger View-Specific Events in Pluggable Views

View-specific events in pluggable views are generally fired when users perform an action in the view. In addition to updating the visual user interface of the view, theses events publish messages that can be wired to create interactions with other apps in a workspace app. They do *not* trigger interactions in other views included in the same basic app.

To provide view-specific events, your pluggable view implementation class must:

- Trigger events with the `Presto.view.trigger(element, event, data)` method. See "Triggering an Event" on page 1060 for an example.
- Declare the names of supported events in the `Presto.view.register(config)` method. See "Registering the Event" on page 1061 for an example.

Triggering an Event

This example is part of a pie view based on the D3js library. For the completed example, see "Example: Complete D3 Pie with View-Specific and DataTable Events" on page 1063. In this example, the view must respond to user clicks in pie slices.

To trigger an event in a pluggable view class, you add an event handler that calls `Presto.view.trigger` to fire this event. In this example, the `onClick` handler for a click event is defined and then registered for each slice of the pie in the `draw` method:

```
   self.draw = function(dataTable, config){
      config = config || self.config;
      var el = $(selector),
onClick = function(event){
//update user interface
//trigger view-specific event
var eventData = {};
eventData[props.label] = d3.select(this).attr("name");
eventData[props.series] = event.value;
Presto.view.trigger(el, 'click', eventData);
return false;
},  ...
      arcs.append("path")
        .attr("fill", function(d, i) {
          return color(i);
        })
        .attr("d", arc)
        .attr("name", function(d, i){
          return labels[i];
        })
.on('click', onClick);
```

The handler typically updates the view user interface in an appropriate manner for this type of visualization. Visual updates must handle both new selection and deselection of rows. To do this properly, views must know the current selected state of rows which is automatically handled by the DataTable for the view. For this example, updating the view is discussed in Support DataTable Row Selection Events in Pluggable Views instead.

The handler must also build the event object for this view-specific event. You can include data for any column that has been included in the view during view configuration. Data for columns that have been removed from the view is *not* available.

Once the event data object is built, the handler simply calls `Presto.view.trigger(el,event,data)` with a CSS selector or the actual DOM node for the view container (the node in the app or page where the view is rendered), the name of the event to fire and the event object.

Finally, the view implementation must register this handler for appropriate user actions in the view. In this example, `onClick` is registered as the handler for click events on each slice of the pie.

Registering the Event

You plug view-specific events for a pluggable view into the MashZone NextGen App Framework so they may be used to wire interactions between different apps. The view class must declare the names of all view-specific events in the call to `Presto.view.register`, such as this example:

```
...
  Presto.view.register({
    clazz: Sample.d3.Pie,
    lib: "d3-pie",
events: ["click"]
});
}(jQuery));
```

Simply add an `events` property to the configuration object and assign an array with the names of each view-specific event to support.

Support DataTable Row Selection Events in Pluggable Views

Supporting the DataTable `rowselect` event in a pluggable view is a good practice for several reasons:

- It allows pluggable views to participate in synchronization of row selections across multiple views. This feature allows views within one basic app to all react appropriately when users select an element in one view. Most MashZone NextGen built-in views already support synchronization.

  Synchronization also applies to views added directly to workspace apps in some circumstances. See "Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace" on page 1125 for more information.

- The `rowselect` event is always visible to users when wiring interactions in Mashboard. With view-specific events, users may need to perform click events to make them visible for wiring.

- View-specific events publish only the data that is included for the view. The `rowselect` event instead includes data for the entire row. Having all data available is highly useful when wiring apps in workspaces using Mashboard to provide better options for mapping events in different apps.

To support `rowselect` events in the DataTable for your pluggable view, the view must Notify the DataTable of rowselect Events and Listen for rowselect Events from the DataTable. See also "Example: Complete D3 Pie with View-Specific and DataTable Events" on page 1063 for the complete example pluggable D3 Pie view.

Notify the DataTable of rowselect Events

Pluggable views can have their DataTable fire `rowselect` events when users click elements within the view using the `rowClick(`*`rowIndex`*`)` method in the MashZone

NextGenDataTable API. The *rowIndex* parameter identifies which row in the DataTable has been selected by the user.

The DataTable uses this information to update selection state information for both the newly selected row and for the previously selected row, if any. It then fires one or two `rowselect` events:

If	Fire
No row was previously selected	Rowselect for the new selected row with selected = true.
A row was previously selected and the new row is a different row.	■ Rowselect for the previously selected row with selected = false. ■ Rowselect for the new selected row with selected = true.
A row was previously selected and the new row is that same row.	Rowselect for the previously selected row with selected = false.

Each `rowselect` event includes the index for the row, the selected state (true or false) for this row and an object with the data for this entire row.

This example extends the click event handler method for each slice of the custom D3 Pie view shown in Triggering an Event. It makes use of a feature in D3 that provides the index number to the data for any element that is clicked. Using this index, the handler simply calls `rowClick`:

```
...
      onClick = function(event, index){
        //update datatable for rowselect
dataTable.rowClick(index);
        //trigger view-specific event
        var eventData = {};
        eventData[props.label] = d3.select(this).attr("name");
        eventData[props.series] = event.value;
        Presto.view.trigger(el, 'click', eventData);
        return false;
      },
...
```

In many cases, the click handler would also update the user interface to indicate that selection had occurred. To accurately do this, the handler needs to know the previous selection state for each slice.

Since the DataTable tracks the selection status and also fires `rowselect` events that fully update this status for all rows involved, this example delegates the process of updating the user interface to the handler for `rowselect`.

Listen for rowselect Events from the DataTable

Pluggable views can also listen for `rowselect` events from the DataTable to account for user selections made in other views using this DataTable. To do this, the view must implement a handler for the event and then call the `addListener(event-name,handler,scope)` method for the DataTable.

This example adds the `onRowSelect(rowIndex,selected,rowData)` method as the handler that updates the user interface of the view each time a `rowselect` event is fired by the DataTable. The parameters for `onRowSelect` are the properties of the `rowselect` event: the row index, its selected state and the data for the row.

Since user clicks within the view can also trigger `rowselect` events in the DataTable, this method handles both clicks within the view or clicks from other views using this DataTable. See "Notify the DataTable of rowselect Events" on page 1061 for more information on the `rowselect` events the DataTable may fire.

```
...
      //determines selected or deselected color for slice and applies
      //based on selected state from datatable rowselect event
onRowSelect = function(index, selected, row) {
var c = color(index);
if(selected){
c = d3.rgb(c).brighter();
}
d3.select(arcs[0][index]).select('path').attr('fill', c);
},
...
      //add datatable listener for rowselect event
dataTable.addListener('rowselect',onRowSelect,self);
...
```

In this example, the handler updates the color of the slice to a brighter shade if the row is now selected or returns the color for the slice to its original shade if the row is deselected.

Once the handler exists, the view calls `addListener`, to register the handler with the DataTable.

Example: Complete D3 Pie with View-Specific and DataTable Events

This is the completed D3 Pie pluggable view that has both a view-specific `click` event and fully supports (triggers and listens for) DataTable `rowselect` events. This view depends on the D3js third-party library which has been imported into MashZone NextGen as a pluggable library.

Important: MashZone NextGen includes certain "*third party software*" which JackBe licenses from third parties. Pursuant to the JackBe EULA for MashZone NextGen, all MashZone NextGen users are bound by the license terms and conditions of any third party software licenses. You may review these Third Party Licenses at http://documentation.softwareag.com/legal/.

Software AG does not and cannot authorize any use of third party software that is not permitted by these third party software licenses. You may, however, be able to obtain licenses directly from third parties. In addition,

any other software that you add and/or use in connection with MashZone NextGen is subject to its own licensing requirements and may void the terms of the EULA for MashZone NextGen.

See "Example: Pluggable View Import with Library Dependencies" on page 1081 for an example of the library properties and import process for this example.

```
(function($){
  var configForm = [
    '<div
>',
      '<div>Enter Label:</div><div><select name="label"
 ></select></div>',
      '<div>Enter Series:</div><div><select name="series"
 ></select></div>',
    '</div>'
  ].join("");
  Presto.namespace('Sample.d3');
  Sample.d3.Pie = function(selector, dataTable, config){
    var self = this,
    getData = function(rows, props){
      var i, data = [];
      for(i=0; i < rows.length; i++){
        data.push(rows[i][props.series]);
      }
      return data;
    },
    getLabels = function(rows, props){
      var i, labels = [];
      for(i=0; i < rows.length; i++){
        labels.push(rows[i][props.label]);
      }
      return labels;
    };
    self.config = config || {};
    self.selector = selector;
    self.draw = function(dataTable, config){
      config = config || self.config;
      var el = $(selector),
      onClick = function(event, index){
        //update datatable for rowselect
        dataTable.rowClick(index);
        //trigger view-specific event
        var eventData = {};
        eventData[props.label] = d3.select(this).attr("name");
        eventData[props.series] = event.value;
        Presto.view.trigger(el, 'click', eventData);
        return false;
      },
      //determines selected or deselected color for slice and applies
      //based on selected state from datatable rowselect event
      onRowSelect = function(index, selected, row) {
        var c = color(index);
        if(selected){
          c = d3.rgb(c).brighter();
        }
        d3.select(arcs[0][index]).select('path').attr('fill', c);
      },
      props = config.properties || {},
      width = config.width || el.width() || 400,
      height = (config.height || el.height() || 340),
        rows = Presto.view.getRows(dataTable, config.columns, { flatten: true }),
```

```
    data = getData(rows, props),
    labels = getLabels(rows, props),
    outerRadius = Math.min(width, height) / 3,
    innerRadius = outerRadius * .0,
    color = d3.scale.category20(),
    donut = d3.layout.pie(),
    arc = d3.svg.arc().innerRadius(innerRadius).outerRadius(outerRadius);
    //add datatable listener for rowselect event
    dataTable.addListener('rowselect',onRowSelect,self);
    var vis = d3.select(el[0]).html('')
      .append("svg")
      .data([data])
      .attr("width", width)
      .attr("height", height);
    var arcs = vis.selectAll("g.arc")
      .data(donut)
      .enter().append("g")
      .attr("class", "arc")
      .attr("transform", "translate(" + (width/2) + "," + (height/2) + ")");
    arcs.append("path")
      .attr("fill", function(d, i) {
        return color(i);
      })
      .attr("d", arc)
      .attr("name", function(d, i){
        return labels[i];
      })
      .on('click', onClick);
    arcs.append("text")
      .attr("transform", function(d) {
        return "translate(" + arc.centroid(d) + ")";
      })
      .attr("dy", ".35em")
      .attr("text-anchor", "middle")
      .attr("display", function(d) {
         return Math.abs(d.startAngle - d.endAngle) > (Math.PI / 10) ? null : "none";
      })
      .text(function(d, i) {
        return labels[i] + ' ' + d.value.toFixed(2); |
      });
    arcs.append("title")
      .text(function(d, i) { return labels[i] + " : " + d.value.toFixed(2); });
  };
  self.showConfig = function(dataTable, selector, config){
    config = config || self.config;
    var el = $(selector),
    props = config.properties || {},
    initColumnSelect = function(name, value, type){
      var $sel = el.find('select[name='+name+']');
      $sel.empty();
      config.columns.each(function(col){
        var name = col.name,
        datatype = col['data-type'],
        selected = name == value ? 'selected="selected"' : '';
        if(!type || type == datatype){
          $sel.append('<option value="' + name + '" ' + selected + ' >' + name + '</option>');
        }
      });
    };
    self.configForm = el;
    self.config = config;
    el.html(configForm);
    if(el){
```

```
        initColumnSelect('label', props.label);
        initColumnSelect('series', props.series);
      }
    };
    self.getConfig = function(){
      var el = self.configForm, config = self.config || {};
      if(el){
        config.properties = {
          label: el.find('select[name=label]').val(),
          series: el.find('select[name=series]').val()
        }
      }
      return config;
    };
    self.validate = function(){
      var config = self.getConfig(), props;
      if(config){
        props = config.properties;
        if(props.label && props.series){
          return true;
        }
      }
      return false;
    };
  };
  Presto.view.register({
    clazz: Sample.d3.Pie,
    lib: "d3-pie",
    events: [ "click" ]
  });
}(jQuery));
```

CSS Styles, Help and Thumbnails for Pluggable Views

CSS Styles for Pluggable Views

The best practice to maintain a reliable visual style for pluggable views is to define all styles in CSS rules based on classes that are *unique* to the pluggable view implementation. This ensures that other CSS styles that are present in the page, device or app where the view is rendered do not conflict with the visual requirements for your pluggable view.

To implement this:

- Assign a unique CSS class to the container element for the view. This is typically defined in the `draw` and/or `update` methods for the view implementation class.
- Create all styles in a CSS file local to the pluggable view. Typically, this is in the `/css` folder.
- Ensure that all CSS rules are relative to the base CSS class for the container element.

With a class name of `.folder-diagram` on the root node of the view, the CSS rules would look something like this:

```
.folder-diagram { font-family:san-serif; font-size: 12px;
  ...}
.folder-diagram table { border: solid 1px #990000; padding: 3px;
  ... }
```

```
.folder-diagram .root { font-weight: bold; }
.folder-diagram .branch { margin-left: 6px;
  ...}
```

Help Topics for Pluggable Views

The MashZone NextGenView Maker wizard includes a **?** button to provide access to help topics for each view in the View Gallery. To include help for a pluggable view, write the help topic as an HTML page and specify this in the `helpUrl` property in the properties file. For an example, see "Help Topics for Pluggable Views" on page 1067.

Thumbnails for Pluggable Views

MashZone NextGen provides a default thumbnail image to use for pluggable views in the View Gallery:

You can provide your own thumbnail for a pluggable view in the /thumbnail folder of the library or by specifying a URL in the `thumbnailUrl` property in the library's properties file. Thumbnail images must be 72 pixels square. For an example, see "Thumbnails for Pluggable Views" on page 1067.

Configuration Properties for Pluggable Views or Libraries

When you import pluggable views or pluggable libraries, import adds them as pluggable libraries with local resources *by default*. See "Example: Default Library Import with No Properties" on page 1076 for an example.

You can use configuration properties to change this default or to manage other aspects of a pluggable view or pluggable library. Configuration properties can:

- Identify this as a pluggable view and help to integrate it with the MashZone NextGen View Gallery.
- Define dependencies on other libraries and ensure that MashZone NextGen can successfully load this pluggable view or library.
- Identify library resources when all resources are hosted externally.
- Manage ppluggable view and library versions and updates. See "Managing Updates and Library Versions" on page 1085 for more information on this usage.
- Provide other metadata for the view or library.

You set view/library configuration properties in either a *properties file* that you add to the library resources or in an *Apache Ant™ build file*. See "Library Projects Folders and Configuration Files" on page 1073 or "Ant Tasks and Sample Build File to Import Pluggable Views or Libraries" on page 1074 for more information.

See "Pluggable View or Library Configuration Properties" on page 1068 for a complete list of properties that you can use.

For examples of how properties affect imports, see:

- "Example: Default Library Import with No Properties" on page 1076
- "Example: Controlling the Library Import Process" on page 1076
- "Example: Basic Library Import Properties" on page 1077
- "Example: Pluggable View Import Basics" on page 1078
- "Example: Pluggable View Import with Device Compatibility" on page 1080
- "Example: Pluggable View Import with View Gallery Properties" on page 1080
- "Example: Pluggable View Import with Library Dependencies" on page 1081
- "Example: Pluggable Library Import for an Externally Hosted Library" on page 1084

Pluggable View or Library Configuration Properties

id	The ID for this pluggable library or pluggable view. This *must* match the name of the root folder that contains the resources for this view or library and must be a unique library name in this MashZone NextGen Repository. Library IDs should be limited to ASCII letters, numbers and the dash(-) or underscore (_) characters. **Note:** For custom apps, this ID is the library name in the App Spec.
name	The title to display in the MashZone NextGen View Gallery for a pluggable view. Capitalization is unimportant in view titles as they display in a fully capitalized font. View titles should be short. Titles with more than 14 characters may be truncated with a trailing ellipsis (...). The complete title is shown as a tooltip.

description	An optional, short description of the purpose of this pluggable library or pluggable view. For pluggable views, this displays as a tooltip in the View Gallery.
type	Either `view`, for a pluggable view, or omit this property. See "Example: Pluggable View Import Basics" on page 1078, "Example: Pluggable View Import with Device Compatibility" on page 1080, "Example: Pluggable View Import with View Gallery Properties" on page 1080 and "Example: Pluggable View Import with Library Dependencies" on page 1081 for examples.
subtype	For pluggable views, the category in the View Gallery where this pluggable view should be listed. This can be the name of an existing category or a new category name. If you omit this property, pluggable views are listed in the `Pluggable Views` category in the View Gallery. See "Example: Pluggable View Import with View Gallery Properties" on page 1080 for examples.
thumbnailUrl	An optional URL to the image to display in the View Gallery for this pluggable view. This URL may be relative, pointing to an image in folders for this library, or an absolute URL. **Note:** Thumbnail images for pluggable views should be 72 pixels square. If you omit this property, MashZone NextGen will use the image, if any in the /thumbnail folder of the pluggable view. If no /thumbnail folder is present and you omit this property, a default image is used: See "Example: Pluggable View Import with View Gallery Properties" on page 1080 for examples.
helpUrl	A URL to an HTML page with user assistance information to help them configure a view using this pluggable view

	library. This help topic opens in a new window from the ? button in the Create View wizard. This can be a relative URL, within the folders of the pluggable view, or an absolute URL. See "Example: Pluggable View Import Basics" on page 1078 for examples.
version	The version number to use for this pluggable library or pluggable view. If omitted, this defaults to `1`. For more information on library versions, see "Managing Updates and Library Versions" on page 1085.
defaultVersion	Whether this version of the pluggable library or pluggable view is considered the *default version*. This defaults to `true`. Whether a view or library is the current default also determines whether existing views in mashables, mashups or apps use this version of the pluggable view or library. For more information on version control and updates to MashZone NextGen artifacts, see "Managing Updates and Library Versions" on page 1085.
libPath	■ For all pluggable views or for pluggable libraries with local resources that you host in MashZone NextGen, this property is calculated by MashZone NextGen. ■ If this is a pluggable library with resources that are hosted externally, this property is the base URL to prepend to all JavaScript and CSS resources for this library. See "Example: Pluggable Library Import for an Externally Hosted Library" on page 1084 for an example.
js	An optional, comma-separated list of the JavaScript resources to include in this pluggable view or pluggable library. ■ For all pluggable views or for pluggable libraries with local resources that you host in MashZone NextGen, this property is most commonly calculated by MashZone NextGen based on the JavaScript files found in the /js folder of the pluggable view or library. If you want to exclude some files in this folder, you can use this property to list only those files that should be included in MashZone NextGen. ■ For pluggable libraries that are hosted externally, list the specific JavaScript files to include in this library relative

	to the URL in the `libPath` property. See "Example: Pluggable Library Import for an Externally Hosted Library" on page 1084 for an example.
css	An optional, comma-separated list of the CSS resources to include in this pluggable view or pluggable library. ■ For all pluggable views or for pluggable libraries with local resources that you host in MashZone NextGen, this property is most commonly calculated by MashZone NextGen based on the CSS files found in the /css folder of the pluggable view or library. If you want to exclude some files in this folder, you can use this property to list only those files that should be included in MashZone NextGen. ■ For pluggable libraries that are hosted externally, list the specific CSS files to include in this library relative to the URL in the `libPath` property. See "Example: Pluggable Library Import for an Externally Hosted Library" on page 1084 for an example.
dependsOn	If this pluggable library or pluggable view uses bundled libraries or other pluggable libraries, list the IDs of those libraries separated by commas. This information allows MashZone NextGen to ensure that all dependencies are loaded before loading this library. **Important:** MashZone NextGen includes certain "*third party software*" which JackBe licenses from third parties. Pursuant to the JackBe EULA for MashZone NextGen, all MashZone NextGen users are bound by the license terms and conditions of any third party software licenses. You may review these Third Party Licenses at http://documentation.softwareag.com/legal/. Software AG does not and cannot authorize any use of third party software that is not permitted by these third party software licenses. You may, however, be able to obtain licenses directly from third parties. In addition, any other software that you add and/or use in connection with MashZone NextGen is subject to its own licensing requirements and may void the terms of the EULA for MashZone NextGen. See "Example: Pluggable View Import with Library Dependencies" on page 1081 for examples.

loadConfirmation	A boolean expression containing one or more JavaScript namespaces or objects from this library that must exist at runtime to confirm that this library is completely loaded. For example: `MyOrg && MyOrg.MyView` This expression is not required but is *highly* recommended to ensure that libraries for pluggable views or custom apps are properly loaded before rendering. See "Example: Basic Library Import Properties" on page 1077 for an example.
desktop	Whether this pluggable library or pluggable view is compatible with desktop devices. Defaults to `true`. Views created from this pluggable view inherit this device compatibility. This, in turn affects device compatibility for apps create from these views. See "Example: Pluggable View Import with Device Compatibility" on page 1080 for more information.
mobile	Whether this pluggable library or pluggable view is compatible with mobile devices. Defaults to `true`. Views created from this pluggable view inherit this device compatibility. This, in turn affects device compatibility for apps create from these views. See "Example: Pluggable View Import with Device Compatibility" on page 1080 for more information.
hidden	Whether this pluggable library or pluggable view should be hidden in the App Editor, and thus not available for use in custom apps. This defaults to `false` meaning the library will be visible in the App Editor. Change this to `true` to keep this pluggable view or library off the list of available, named libraries that users can add to custom apps.
createdBy	An optional username for the user who should be registered as the owner of this pluggable view or pluggable library. If omitted, MashZone NextGen uses the username you specify when you register and upload the pluggable view or library as the owner.

Import Pluggable Views and Libraries

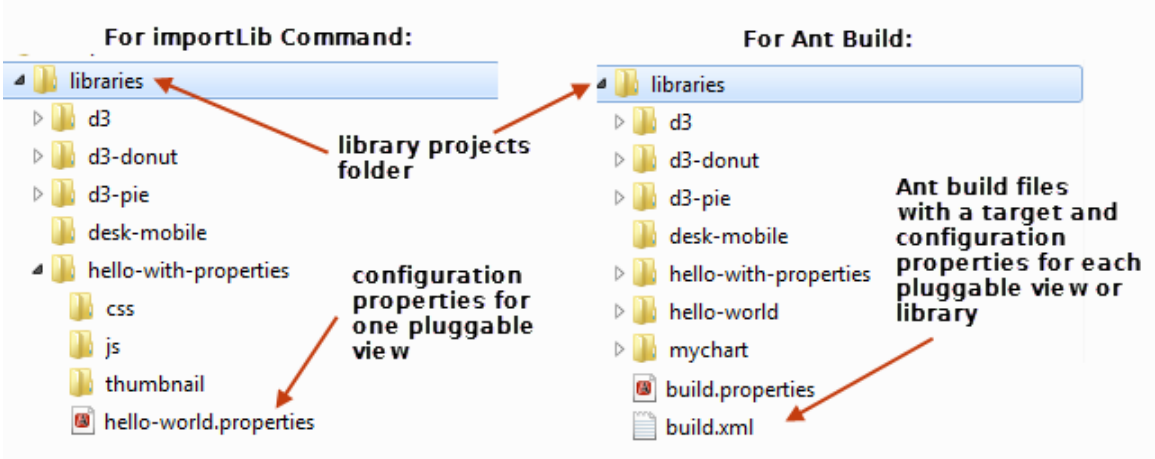
You can import pluggable views and libraries individually using the MashZone NextGen`importLib` command. Or use the Apache Ant™ build tool if that is available in your development environment.

In either case, you must set up Library Projects Folders and Configuration Files for your pluggable views and pluggable libraries. For examples of importing with different configuration information, see:

- "Example: Default Library Import with No Properties" on page 1076
- "Example: Controlling the Library Import Process" on page 1076
- "Example: Basic Library Import Properties" on page 1077
- "Example: Pluggable View Import Basics" on page 1078
- "Example: Pluggable View Import with Device Compatibility" on page 1080
- "Example: Pluggable View Import with View Gallery Properties" on page 1080
- "Example: Pluggable View Import with Library Dependencies" on page 1081
- "Finding Bundled or Other Pluggable Libraries" on page 1083
- "Example: Pluggable Library Import for an Externally Hosted Library" on page 1084

Library Projects Folders and Configuration Files

The configuration files you need are somewhat different if you are importing pluggable views and libraries using the MashZone NextGen`importLib` command or using Ant:



You need a folder, called the root library folder, for each pluggable view or library. The root library folder name is also the ID for the pluggable library or view. Subfolders hold the different types of resources. For configuration files:

- *To import with the* `importLib`*command*, each pluggable view or library has its own properties file in the root library folder with import configuration information, such as the hell-world.properties file shown in this example.

  See "Configuration Properties for Pluggable Views or Libraries" on page 1067 for information on all the properties you can use for a pluggable view or pluggable library. See "Import Pluggable Library/View Examples" on page 1076 for examples of property files.

- *To import with Ant*, you define a target to import each pluggable view or library and set configuration properties in the custom task for that target. See "Configuration Properties for Pluggable Views or Libraries" on page 1067 for information on all the properties you can use for a pluggable view or pluggable library. See "Ant Tasks and Sample Build File to Import Pluggable Views or Libraries" on page 1074 for information on setting up an Ant build file for view or library imports.

Ant Tasks and Sample Build File to Import Pluggable Views or Libraries

If you are using Ant to import or export pluggable views or libraries, you must add a <taskdef> in the Ant build.xml file to handle pluggable library or view imports, and optionally another to handle exports, such as this example:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Import/export pluggable views and libraries" default="all" basedir="." >
  <!-- custom properties to your environment -->
  <property file="build.properties"/>
  <!-- path to library project folder -->
  <property name="libs.dir" value= "${basedir}" />
  <!-- path to JAR for custom tasks, presto.dir is installation directory,
       set in build.properties -->
  <property name="prestocli.jar"
    value="${presto.dir}/prestocli/dist/prestocli.jar"/>
 <!--MashZone NextGencustom tasks to import/export pluggable libraries -->
  <taskdef name= "presto.lib.import"
    classname="com.jackbe.jbp.sas.cli.ant.ImportLib"
    classpath="${prestocli.jar}" />
  <taskdef name="presto.lib.export"
   classname="com.jackbe.jbp.sas.cli.ant.ExportLib"
   classpath="${prestocli.jar}" />
  <!-- targets to import libraries in this project -->
  <target name="lib-import-basic-properties">
    <presto.lib.import
      id="mylibrary" name="My Common Library"
      description="My common library for all views"
      dir="${libs.dir}/mylibrary"
      loadConfirmation="Sample.library.MyCommonLibrary"
      overwrite="true" continueOnError="true" />
  </target>
  <!-- 3rd party library hosted externally -->
  <target name="external-lib-import">
    <presto.lib.import
      id="reportgrid" name="Report Grid"
      description="Report Grid" loadConfirmation="ReportGrid"
      libPath="http://api.reportgrid.com/js"
      js="reportgrid-charts.js" />
  </target>
  <!-- 3rd party library hosted inMashZone NextGen-->
  <target name="d3-lib-import">
```

```
    <presto.lib.import
      id="d3" name="D3"
      description="D3 - Data Driven Documents" dir="${libs.dir}/d3"
      loadConfirmation="window.d3" js="js/d3.v2.js" />
  </target>
  <!-- basic view library import -->
  <target name="minimal-view-import">
    <presto.lib.import
      id="mysample" name="My Sample View"
      description="minimal pluggable view" dir="${libs.dir}/mysample"
      type="view" loadConfirmation="Sample.view.MySampleView"/>
  </target>
  <!-- view with dependency on 3rd party and built-in libraries -->
  <target name="pie-view-import-d3-dependency" >
    <presto.lib.import
      id="d3-pie" name="D3 Pie Chart"
      description="Sample D3 Pie Chart" dir="${libs.dir}/d3-pie"
      loadConfirmation="Sample.d3.Pie" type="view"
      dependsOn="d3,presto-core" />
  </target>
  <!-- target to import D3 and all dependent views -->
  <target name="import-d3-views" depends="d3-lib-import,
    pie-view-import-d3-dependency" />
  <target name="all" depends="import-d3-views,minimal-view-import,
    external-lib-import,lib-import-basic-properties" />
...
</project>
```

Then create a separate target to import each pluggable view or library and use the `<presto.lib.import>` task to perform the import. You can set any Configuration Properties for Pluggable Views or Libraries in the `<presto.lib.import>` task . In addition, you can also set three boolean flags to control aspects of how `importLib` is run:

- `verbose` = -v option, to control how much information is logged.
- `overwrite` = -o option, to control whether this import overwrites the existing version, if any, of this pluggable library.
- `continueOnError` = -c option, to control whether the build should continue if an error occurs during the build.

By default, all these flags are `false`. See "Example: Controlling the Library Import Process" on page 1076 for more information on these flags.

Finally, you need to set the following properties in build.properties to fit your environment:

```
# administration account username
presto.username=Administrator
# administration account password
presto.password=manage
# host:port to Mashup Server
presto.hostname=localhost:8080
# path toMashZone NextGen installation folder
presto.dir=/users/myname/Presto3.5
```

Import Pluggable Library/View Examples

The examples in this section all use the `importLib` command to import pluggable views or pluggable libraries. They use individual property files to set import configuration properties.

The configuration properties you use in an Ant build file and the effects of importing these libraries are identical.

Example: Default Library Import with No Properties

This example in a Windows environment imports a pluggable library with the local library resources shown below. No configuration properties are set. The library is registered with the MashZone NextGen Server at the default URl (`http://localhost:8080/presto/edge/api`) using the default MashZone NextGen admininstrator account.

Files:	`c:\libraries`
	`  -\mysample`
	`  -\js\lib.js`
	`  -\css\lis.css`

Properties:	No properties file.
Steps:	1. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder for the MashZone NextGen Server 2. Enter this command: `c:\Presto3.5\prestocli\bin> padmin importLib -d c:\libraries\mysample -u Administrator -w manage`

Result:	Because the library has no configuration properties set, this library is automatically imported as a pluggable library with an ID of `mysample`. The two resource files found in the library are uploaded to MashZone NextGen as part of version 1 of this library:
	`/presto/files/system/lib/mysample/1/js/lib.js`
	`/presto/files/system/lib/mysample/1/css/lib.css`
	The library will be listed as a named library available for custom apps in the App Editor. It is also marked as both desktop- and mobile-compatible. It has no dependencies on other libraries.

Example: Controlling the Library Import Process

This example is identical to the library in Example: Default Library Import with No Properties but adds to control how the import process runs.

Files:	`c:\libraries`
	`  -\mysample`
	`  -\js\lib.js`
	`  -\css\lis.css`

Properties:	No properties file.
Steps:	1. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder for the MashZone NextGen Server 2. Enter this command: `c:\Presto3.5\prestocli\bin> padmin importLib -d c:\libraries\mysample -c -o -v -u Administrator -w manage`
Result:	The `overwrite` or `-o` flag allows this import to overwrite an existing version of this library and version, if it exists in MashZone NextGen without failing. For examples and more information on version controls for pluggable libraries, see "Managing Updates and Library Versions" on page 1085. The `verbose` or `-v` flag provides more information in the log about the import process. The `continueOnError` or `-c` flag allows the import process to continue if an exception or error is thrown during the import. This is generally useful if you are using Ant to import multiple or all libraries in build.xml or you are using a script to import multiple libraries with `importLib`.

Example: Basic Library Import Properties

This example sets basic properties for a pluggable library, including an ID, a title, a description and a `loadConfirmation` property.

The `id` property is *required* and its value *must* match the name of the root library folder. The `name` and `description` properties provide general information about the library for MashZone NextGen administrators who manage pluggable libraries or developers who may see this in the App Editor.

MashZone NextGen uses the `loadConfirmation` property to confirm when this library is completely loaded when it is used in pluggable views or custom apps. It is not required, but it is *highly recommended*. In this example, the value of `loadConfirmation` is simply the class name for this library. The value generally needs to be a boolean expression of the classes or objects that must be present before the library can be used successfully.

Files:	`c:\libraries`
	`  -\mylibrary`
	`  -mylibrary.properties`
	`  -\js\lib.js`

	-\css\lib.css

Properties:	#properties for mylibrary
	id=mylibrary
	name=My Common Library
	description=My common library for all views
	loadConfirmation=Sample.library.MyCommonLibrary

Steps:	1. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder for the MashZone NextGen Server 2. Enter this command: `c:\Presto3.5\prestocli\bin> padmin importLib -d c:\libraries\mylibrary -u Administrator -w manage`

Result:	Based on this configuration, the library is registered as:
	■ A pluggable library with an ID of `mylibrary` and the specified description
	■ Version 1
	■ Both desktop- and mobile-compatible
	■ With no dependencies
	■ Fully loaded only when an object named `Sample.library.MyCommonLibrary` is present in the window
	■ Visible in the App Editor for use in custom apps
	■ With three resources uploaded and hosted in the MashZone NextGen Server as:
	  ■ /system/lib/mylibrary/1/mylibrary.properties (only if imported with `importLib`)
	  ■ /system/lib/mylibrary/1/js/lib.js
	  ■ /system/lib/mylibrary/1/css/lib.css
	■ Owned by the `Administrator` account
	Note: Although you can access library files that are hosted in MashZone NextGen directly using the appropriate URL, for pluggable views or custom apps you only need to add configuration to identify the pluggable library as a dependency. MashZone NextGen handles all the requirements to load all the library's resources for you.

Example: Pluggable View Import Basics

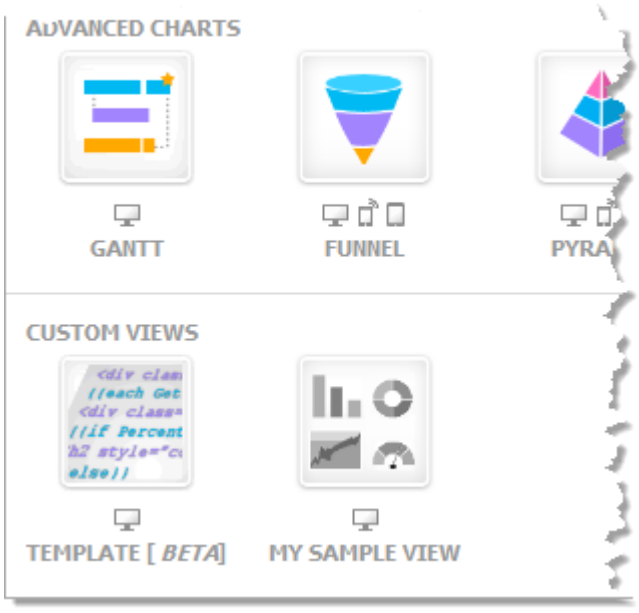
To register a pluggable view, you must include the `type` property with a value of `view`.

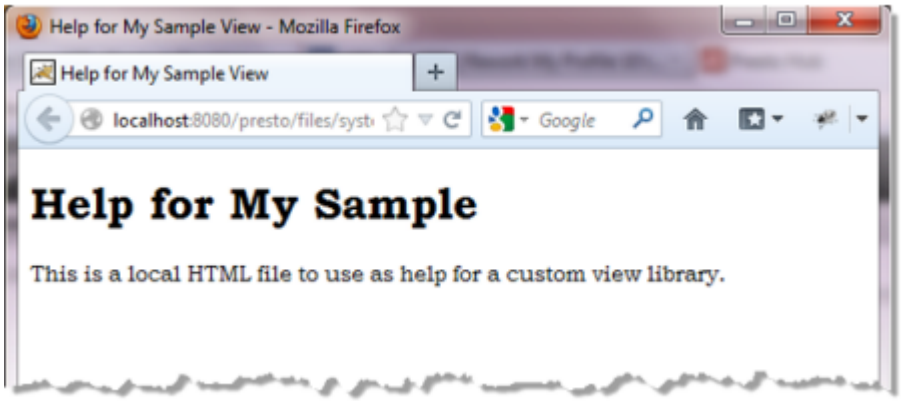
This example also includes the `helpUrl` property to provide a user assistance topic for users when they choose to add this pluggable view to a mashable or mashup. The

helpUrl property for this example points to a local HTML resource in the pluggable view library, although you can point to external resources.

Files:	`c:\libraries`
	`  -\mychart`
	`  -mychart.properties`
	`  -\js\lib.js`
	`  -\css\lib.css`
	`  -\html\help.html`

Properties:	`#properties for mychart`
	`id=mychart`
	`name=My Sample View`
	`description=Sample chart view`
	`loadConfirmation=Sample.view.MyChart`
	`type=view`
	`helpUrl=html/help.html`

Steps:	1. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder for the MashZone NextGen Server 2. Enter this command: `c:\Presto3.5\prestocli\bin> padmin importLib -d c:\libraries\mychart -u Administrator -w manage`
Result:	Based on this configuration, the library is registered as a pluggable view library. It will also be visible in the App Editor for use in custom apps. This pluggable view appears in the MashZone NextGen View Gallery in the `Pluggable Views` category with `My Sample View` as its name. It uses the default thumbnail image for pluggable views: 

	If users click the *?* button while they are configuring this view for a mashable or mashup, the mychart/html/help.html file opens in a smaller separate window in most browsers as user assistance for this pluggable view. 

Example: Pluggable View Import with Device Compatibility

This pluggable view is identical to the pluggable view in Example: Pluggable View Import Basics, but it uses the `mobile` property to indicate that this pluggable view is *not* mobile compatible:

Properties:	`#properties for mychart`
	`id=mychart`
	`name=My Sample View`
	`description=Sample chart view`
	`loadConfirmation=Sample.view.MyChart`
	`type=view`
	`helpUrl=html/help.html` **`mobile=false`**

Result:	With `mobile` set to false, this view is generally not visible in mobile devices. See "About Desktop and Mobile Compatibility for Apps" on page 1111 for more information.

Example: Pluggable View Import with View Gallery Properties

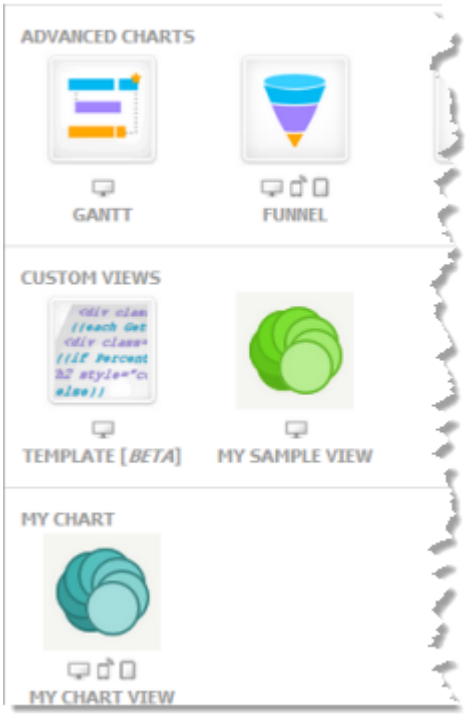
This example of a pluggable view is identical to the pluggable view in Example: Pluggable View Import Basics, but it uses two properties to customize where the pluggable view appears in the MashZone NextGen View Gallery for users and what image is used as the thumbnail.

The `subtype` property identifies the category (the group) where a pluggable view appears in the Gallery. This can be a category you define or one of the built-in categories in the MashZone NextGen View Gallery.

For the thumbnail, you can simply include the image in a /thumbnail folder within the pluggable view resources or use the `thumbnailUrl` property.

Files:	c:\libraries
	  -\mychart
	  -mychart.properties
	  -\js\lib.js
	  -\css\lis.css
	  -\html\help.html
	  -thumbnail\mychart.png

Properties:	#properties for mychart
	id=mychart
	name=My Chart View
	description=Sample chart view
	loadConfirmation=Sample.view.MyChart
	type=view **subtype=My Chart**
	helpUrl=html/help.html **thumbnailUrl=thumbnail/mychart.png**

Result:	Once you import this pluggable view library, it appears in the MashZone NextGen View Gallery in a new categoy using the specified thumbnail image, such as this: 

Example: Pluggable View Import with Library Dependencies

Pluggable views and custom apps can both use libraries bundled in MashZone NextGen or use pluggable libraries that have been imported to MashZone NextGen. (These pluggable libraries may be directly hosted in MashZone NextGen or may be hosted externally.) Library dependencies are defined in the `dependsOn` property which MashZone NextGen uses to automatically load required libraries and handle duplicate dependencies, if any.

Important: MashZone NextGen includes certain "*third party software*" which JackBe licenses from third parties. Pursuant to the JackBe EULA for MashZone NextGen, all MashZone NextGen users are bound by the license terms and conditions of any third party software licenses. You may review these Third Party Licenses at http://documentation.softwareag.com/legal/.

Software AG does not and cannot authorize any use of third party software that is not permitted by these third party software licenses. You may, however, be able to obtain licenses directly from third parties. In addition, any other software that you add and/or use in connection with MashZone NextGen is subject to its own licensing requirements and may void the terms of the EULA for MashZone NextGen.

This example defines one pluggable library for D3js that supports a variety of diagrams and charts and also defines a pluggable view for MashZone NextGen that implements a D3 bar chart.

The resources and properties for the D3js pluggable library are:

Files:	`c:\libraries\d3`
	`  - d3.properties`
	`  - \js`
	`  - \d3.v2.js`

Properties:	`#properties for D3 library`
	`id=d3`
	`name=D3 Diagrams`
	`description=D3js - Data Driven Documents`
	`loadConfirmation=window.d3`

Steps:	1. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder for the MashZone NextGen Server 2. Enter this command: `c:\Presto3.5\prestocli\bin> padmin importLib -d c:\libraries\d3 -u Administrator -w manage`
Result:	Based on this configuration, the library is registered as: ■ A pluggable library with an ID of `d3` and the specified description ■ Version 1 ■ Both desktop- and mobile-compatible ■ With no dependencies ■ Fully loaded only when an object named `Sample.library.MyCommonLibrary` is present in the window ■ Visible in the App Editor for use in custom apps

	■ One resource uploaded and hosted in the MashZone NextGen Server as /system/lib/d3/1/d3.v2.js. ■ Owned by the `Administrator` account

The properties for the pluggable view library that depends on the D3 library to create a bar chart are:

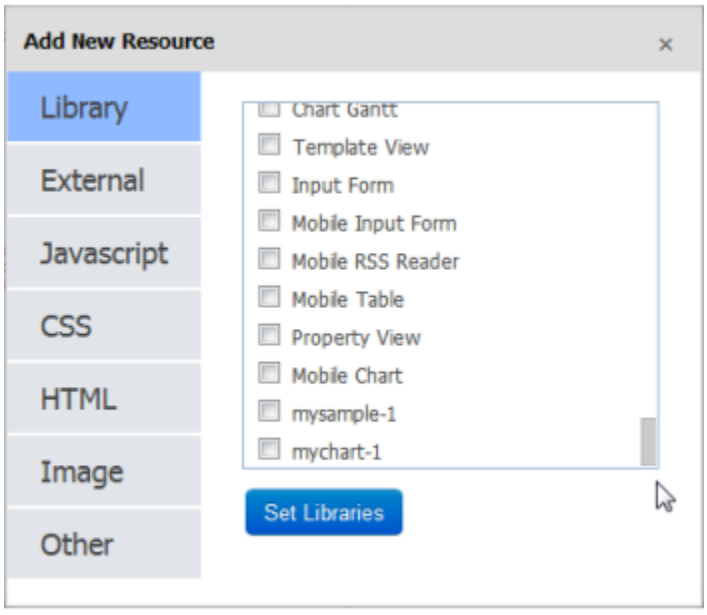
Properties:	`#properties for D3 pie chart view`
	`id=d3-pie`
	`name=D3 Pie Chart`
	`description=D3 pie chart view`
	`type=view`
	`helpUrl=html/d3pie.html`
	`loadConfirmation=Sample.d3.pie` **`dependsOn=d3`**

This example has a single dependency. Multiple dependencies are allowed, simply list each library ID separated with commas. See also "Finding Bundled or Other Pluggable Libraries" on page 1083.

Finding Bundled or Other Pluggable Libraries

You can also use third-party libraries that are bundled in MashZone NextGen in your pluggable views. Or find other pluggable libraries that have already been added to MashZone NextGen.

To find a list of available libraries, open an custom app (or start a new one) in the App Editor. Click **Add** and select the **Library** list. This list shows all libraries bundled in MashZone NextGen plus any imported pluggable libraries that have not been hidden.



This example defines a pluggable view library that uses the bundled jQuery Fancy Zoom plug-in:

Properties:	`#properties for another sample view`
	`id=another`
	`name=Another View`
	`description=View that uses jQuery bundled library`
	`type=view`
	`loadConfirmation=Sample.Another`
	`dependsOn=jquery-plugin-fancyzoom`

Example: Pluggable Library Import for an Externally Hosted Library

Pluggable libraries can also point to third-party libraries that are hosted externally.

Important: MashZone NextGen includes certain "*third party software*" which JackBe licenses from third parties. Pursuant to the JackBe EULA for MashZone NextGen, all MashZone NextGen users are bound by the license terms and conditions of any third party software licenses. You may review these Third Party Licenses at http://documentation.softwareag.com/legal/.

Software AG does not and cannot authorize any use of third party software that is not permitted by these third party software licenses. You may, however, be able to obtain licenses directly from third parties. In addition, any other software that you add and/or use in connection with MashZone NextGen is subject to its own licensing requirements and may void the terms of the EULA for MashZone NextGen.

This example shows the properties to import a pluggable library for the Report Grid charts library, which is hosted externally. There are no resources, so a folder for the library is only required to hold a properties file if you are using `libImport`.

You use the `libPath` property to define the base URL to all resources. Use the `js` and `css` properties to define relative paths to the specific external resources for this library. In this example, only one JavaScript file is used, but you can point to multiple files in both properties.

Files:	`c:\libraries\reportgrid` `  - reportgrid.properties`

Properties:	`#properties for reportgrid library`
	`id=reportgrid`
	`name=Report Grid`
	`description=Report Grid charts`
	`loadConfirmation=ReportGrid` **`libPath=http://api.reportgrid.com/js  js=reportgri`**

Steps:	1. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder for the MashZone NextGen Server 2. Enter this command: `c:\Presto3.5\prestocli\bin> padmin importLib -d c:\libraries\reportgrid -u Administrator -w manage`
Result:	In this case, MashZone NextGen doesn't actually upload any resources except the reportgrid.properties file if you use the

	`importLib` command. The library will be visible in the App Editor and can appear as a dependency for pluggable view libraries.

Managing Updates and Library Versions

Versions for pluggable libraries and pluggable views affect the dependencies between libraries *and* the dependencies for views and the apps that include those views. See "Library Versions and Dependencies" on page 1085 for an overview of these interactions.

There are two significantly different cases for how you manage updates to pluggable libraries or pluggable views and their versions:

- Minor, backwards-compatible updates to an existing library or view.

  With backwards-compatible fixes, you can update the existing library or view. For instructions and examples, see:

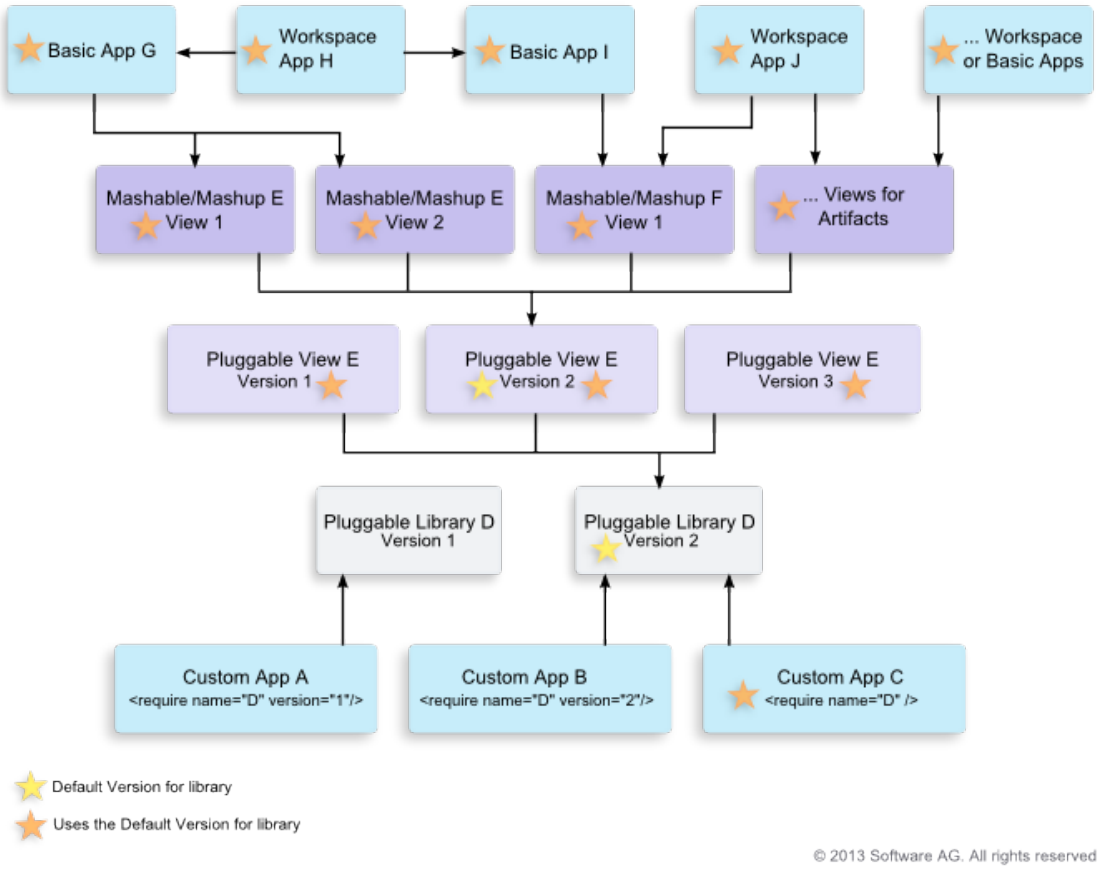
  - "Increment an Existing Library Version" on page 1087
  - "Overwrite an Existing Library" on page 1088

- Upgrades to new releases for third-party libraries or major updates to an existing pluggable library or pluggable view.

  With updates that are not backwards compatible or updates where you want to manage the upgrade process, you must import the updated library with a new library ID. See "Handle New Library Releases or Major Updates" on page 1090 for instructions.

Library Versions and Dependencies

When multiple versions of a pluggable library or pluggable view are present, MashZone NextGen treats one version as the current *default version*. This flag plays a part in determining dependencies for pluggable views, views, and basic, custom or workspace apps as shown in the following diagram:



- *Custom apps* define dependencies on pluggable libraries in the <require> tag in their App Spec based on library name (the ID) and the library version. Changes in library versions mean manual updates to custom app Specifications with one exception. If the <require> tag has no version specified, then the custom app always uses the current default version of that library.
- *Pluggable views* may depend on bundled or pluggable libraries and may in turn be dependencies for views and the basic or workspace apps that use these views.

  Pluggable views define their dependencies on other libraries by library ID only. They always use the current default version of a library. Changes to library versions automatically update the dependent pluggable views.
- *Views, basic and workspace apps* always depend on the current default version for the pluggable view that the view was created with. Updates to the version for pluggable views automatically flow through to the views and the apps that include them, although browser caching for resources may obscure this change.

Handle Minor Library Updates

Increment an Existing Library Version

With minor updates to an existing pluggable library or pluggable view that are backwards-compatible, the best practice is to import updates to the existing pluggable library or view with a new version number. Changing the version number for a pluggable library ensures browsers will retrieve the updated resources rather than using cached versions (the version number is part of the URL for each resource).

Note: If updates to a pluggable library or view are *not* backwards compatible, library updates should be handled as a new library with a new ID. See "Handle New Library Releases or Major Updates" on page 1090 for more information.

To handle updates as a new version of an existing pluggable library or view

1. "Export Resources for an Existing Pluggable View or Library for Updates" on page 1089. This downloads all the resources for that library or view completely expanded in a directory of your choice.
2. Update the resource code as needed.
3. Increment the version number in the properties file for the library or in the <target> for the library in the Ant build.xml file. For example in a properties file:

```
#properties for mysample pluggable view library
 id=mysample
 name=My Sample View
 description=A sample view with help
 type=view
 helpUrl=html/mysample.html
version=2
 loadConfirmation=Sample.SampleView
```

4. Import the update using the `importLib` command or your Ant build.

   For `importLib`:

   a. Open a command or terminal window and move to the *MashZoneNG-install* / prestocli/bin folder for the MashZone NextGen Server

   b. Enter this command:

   ```
   padmin importLib -d path-to-library-root-folder -u username -w password
   ```

   If the MashZone NextGen Server is remote or is not running in Tomcat using the default port, you must also include `-l` *`url-to-mashup-server`* in the command.

The updated resources are imported with a new path for that version. Resources for the existing version remain untouched. With these library resources, for example:

```
c:\libraries\mysample\1
  - mysample.properties
  - /js
```

```
    - lib.js
  - /css
    - lib.css
  - /html
    -mysample.html
  - /thumbnail
    - mysample.png
```

MashZone NextGen would add the following resources for version 2:

```
/presto/files/system/lib/mysample/2/mysample.properties
/presto/files/system/lib/mysample/2/js/lib.js
/presto/files/system/lib/mysample/2/css/lib.css
/presto/files/system/lib/mysample/2/html/mysample.html
```

This new version becomes the default version for the pluggable library or view, so you see immediate changes in views or apps that are dependent on this library. You should also see this version as the default version in the Admin Console.

Overwrite an Existing Library

With minor updates that are backwards-compatible with an existing pluggable library or pluggable view, you can simply overwrite the existing version.

Note: With no change in the library version, the updated resources may not be immediately visible in dependent views or apps because browsers are using cached versions of these resources.

To overwrite an existing library version with updates

1. Export Resources for an Existing Pluggable View or Library for Updates. This downloads all the resources for that pluggable library or view completely expanded in a directory of your choice.
2. Update the resources as needed.
3. Make sure that the version number for this library in its properties file or in the `<target>` in build.xml matches the version number from the download.
4. If you use an Ant build to import this library, make sure the `overwirte` property is set in the `<presto.lib.import>` task for this library's `<target>`. For example:

```
<target name="view-import-mysample">
 <presto.lib.import
  dir="${libs.dir}/mysample"
  id="mysample"
  name="My Sample View"
  description="A sample view with help"
  type="view"
  helpUrl="html/mysample.html"
version="1"
overwrite="true"
  loadConfirmation="Sample.SampleView" />
</target>
```

   Then run the Ant build for this target:

   a. Open a command or terminal window and move to the project folder with `build.xml` for your pluggable libraries and views.

b. Enter the Ant command with the target name for this library. For example:

```
c:\projects\libraries> ant view-import-mysample
```

5. If you use the `importLib` command, *include the* `-o` *option* in the command to overwrite the existing version. For example:

```
padmin importLib -d c:\libraries\mysample\1 -o -u Administrator -w manage
```

As with the `exportLib` command, you may need to specify the URL to the MashZone NextGen Server or provide other options.

Your updates automatically affect all views in mashables or mashups that use this library, any basic or workspace apps in MashZone NextGen Hub that include those views and any custom apps in MashZone NextGen Hub that use this library, *but* the updates may not be immediately visible because of browser caching.

Export Resources for an Existing Pluggable View or Library for Updates

Download the existing resources for the pluggable library or view that you want to update to a folder using the `exportLib` command:

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter this command:

```
padmin exportLib -q "ids=library-id" -d path-to-output-directory -u username -w password
```

Supplying the ID for the library you need to update and the full path to a folder where the resources for this library should be placed.

> **Note:** If the MashZone NextGen Server is remote or is not running in Tomcat using the default port, you must also include `-l` *url-to-mashup-server* in the command.

This downloads all resources for the current default version of the library you requested, starting from the root library folder, to the directory you specified. For example:

Figure 6. Export a Library for Update

With this command in a Windows environment:

```
c:\Presto3.5\prestocli\bin>padmin exportLib -q "ids=mysample" -d c:\libraries -u Administrato
```

If the `mysample` library is a pluggable view and the default current version is 1, the folders and resources that are exported would look something like this:

```
c:\libraries\mysample\1
  - mysample.properties
  - /js
    - lib.js
  - /css
    - lib.css
  - /html
    - mysample.html
  - /thumbnail
    - mysample.png
```

Notice that the version number becomes a folder in the path for the library's resources.

Handle New Library Releases or Major Updates

When updates to custom or third party libraries are far reaching or not backwards compatible with existing versions, updates can cause errors in the dependent pluggable views, views or apps that use those views. See "Library Versions and Dependencies" on page 1085 for more information on library dependencies.

The best practice, in this case, is to import new releases or major updates as a new library, using a different library ID. This allows you to manage how dependent pluggable views, views and apps migrate to this new release or update.

For new releases of third-party libraries, you can simply download the library and add it to MashZone NextGen or add a new library that points to the externally hosted library. See "Creating Pluggable Views or Libraries" on page 1045 for the basic steps. Then update any pluggable views, if any, that should use this new release.

For major updates to pluggable views or pluggable libraries with local code for your organization:

1. Export Resources for an Existing Pluggable View or Library for Updates for the existing version.

   This downloads all resources for the current default version of the pluggable library you requested.

2. Change the root library name to a new ID and update the properties file or `<target>` in the Ant build.xml file to use this new ID.

3. Update the code as needed.

4. Then import the new pluggable library or view with the `importLib` command or your Ant build.

   For `importLib`:

   a. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.

   b. Enter this command:

   ```
   padmin importLib -d path-to-library-root-folder -o -u username -w password
   ```

Add views based on this new pluggable view library and update apps, as needed.

10 Apps and Workspaces

- Create a Basic App 1093
- Publishing, Managing, Sharing and Using Apps 1101
- Edit App Properties 1104
- About Desktop and Mobile Compatibility for Apps 1111
- Title Bars and Toolbar Buttons in Apps 1112
- Publish Apps to the AppDepot 1114
- Create Workspace Apps with Mashboard 1116
- Working in the MashZone NextGen Enterprise AppDepot 1186
- Managing Pending Apps in the AppDepot 1187
- About MashZone NextGen Mobile 1188
- Working in MashZone NextGen Mobile Apps 1189

Apps allow you to display or work with information from MashZone NextGen mashups or mashable information sources in many different destinations and devices. You can work with apps in the AppDepot or the MashZone NextGen Mobile apps, in MashZone NextGen Hub or in other destinations such as portals or SharePoint.

The types of apps that you may create or work with include:

- **Basic Apps and Gadgets**

  You create basic apps from one MashZone NextGen mashup or mashable using one or more views that have been created for that mashup or mashable. Based on their views, basic apps can be used in desktop or mobile destinations. See "Create a Basic App" on page 1093 for instructions.

  In most cases, basic apps display information in fairly common views such as a table, maps or charts. For apps with multiple views, events in one view also update the other views. For example, selecting a row in a table may highlight the corresponding point on a map. Basic apps can also be added to workspaces in Mashboard.

- **Workspace Apps**

  You can also collect apps into *workspaces* with a particular focus or simply organize apps in a useful way. Workspaces contain one or more apps organized in a particular layout, such as a grid or separate tabs. Workspaces can also contain *gadgets* which wrap apps or other content from external sites such as videos or other types of media in your organization.

  You can also *wire* interactions between apps in workspaces so that they work together, as a composition.

  > **Note:** Wiring interactions is also known as *inter-App communication*. It allows events in one app, such as clicking something, to send information to other apps in the workspace that they use to update their data or perform some further action.

  You create workspaces in Mashboard. See "Create Workspace Apps with Mashboard" on page 1116 for more information.

  Workspaces *are* also apps. You can publish workspaces and then find or use them in the AppDepot, the MashZone NextGen Mobile apps or other destinations, just as you do with individual apps.

- **Custom Apps**

  MashZone NextGen developers or administrators can also create *custom apps* that use MashZone NextGen mashups or mashables in custom or more complex views. They can start from basic apps and then extend them for a customized look or feel. Or they can create custom apps from scratch.

  Common examples of custom apps include apps that require additional interaction, such as forms, or mashup/mashable responses that need a unique or custom look and feel.

For custom apps based on a basic app, developers create a basic app and then extend it in the App Editor. See "Customize a Basic App or View" on page 1198 for instructions.

For fully custom apps, developers start in the App Editor. See "Create Fully Custom Apps in the App Editor" on page 1225 for instructions. See also"Custom Mobile App Requirements" on page 1193

Once you or other users have created apps, you can work them much as you do with other MashZone NextGen artifacts. You can embed them or publish them to other destinations. Or publish them to the AppDepot, for desktop access, and the MashZone NextGen Mobile apps, for mobile access. For instructions, see

- "Publishing, Managing, Sharing and Using Apps" on page 1101
- "Publish Apps to the AppDepot" on page 1114
- "Working in the MashZone NextGen Enterprise AppDepot" on page 1186
- "Working in MashZone NextGen Mobile Apps" on page 1189

Create a Basic App

You create basic apps from one MashZone NextGen mashup or mashable information source and include one or more views for that mashup or mashable. You can use basic apps by themselves or in workspaces, which you create in Mashboard.

Note: MashZone NextGen developers can also create custom apps by customizing basic apps or create custom apps from scratch that use several mashups or mashables and other capabilities.

To create a basic app

1. Open the mashup or mashable you want to use, if needed, and run the mashup or mashable to preview response data. See "Run and Preview Mashable/Mashup Data" on page 177 for instructions.
2. If the mashup or mashable has no views, add one or more views. See "Add Views to Mashables and Mashups" on page 901 for instructions.

   Note: If you do not have permissions to add a view, or the view you want to use is not available, you must ask the owner of this mashup or mashable or a MashZone NextGen administrator to add the view you need.

3. Click **Create > App** and follow the steps in the App Maker wizard:
   a. Configure App Input Parameters
   b. Select and Arrange Views
   c. Preview and Finalize the Presentation

d. Save the App

Configure App Input Parameters

If the mashup or mashable has input parameters, you chose how these inputs should be handled for the app in the **Configure Inputs** step.

For desktop apps *only*, you can choose one of two methods to open the form where users set input parameters:

- **Dialog** opens the form as a window on top of the app when it first opens. Users can reopen this form to change inputs using the **App Tools** button in the app's title bar.

  > **Note:** Mobile apps always place the input form in a separate window and provide a button in the title bar to open this window.

- **View** makes the form one view that is always open within the layout of the app.

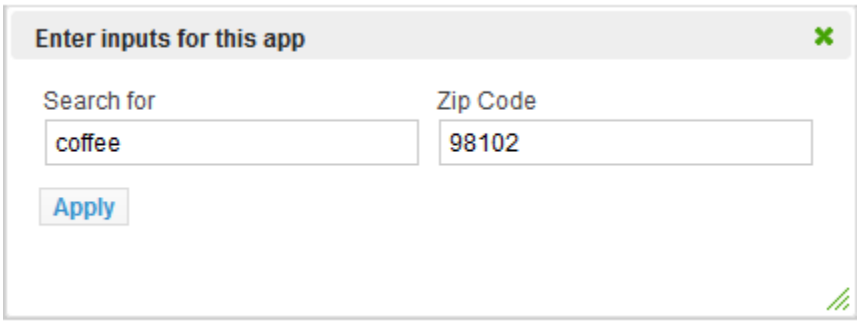
Select a parameter to update how that parameter is handled. There are four patterns for input parameters:

- *Hidden*: for parameters that users should *not* change and do not need to see, such as technical keys necessary to run a mashable. Hidden properties are visible for wiring apps in Mashboard, however.
- *Read-only*: for parameters that users should *not* change but do need to see. Read-only input parameters are visible in app settings in Mashboard, in apps users have saved to Favorites in the AppDepot, and for wiring apps in Mashboard.
- *Indirect*: for parameters whose values are supplied by saved preferences for users in the AppDepot.

  Indirect input parameters are not present in the input form for an app. If all input parameters for an app are indirect, the app has no input form.

- *Direct*: for input parameters that either users supply directly in the input form, at any time when they use this app, or are supplied automatically through wiring in a workspace app.

  Direct input parameters appear in the input form for the app, unless that form has been suppressed in a workspace app. (See "Hiding Input Forms for Apps in a Workspace" on page 1181 for more information.)



Direct input parameters give users control. Users cannot, however, override direct input parameters with their saved preferences for a custom Favorite App in the AppDepot.

Property	Hidden	Read-Only	Indirect	Direct
Editable	no	no	yes	yes
Visible	no	yes	no	yes
Required	---	---	optional	optional
Default Value	required	required	optional	optional
Label defaults to the input parameter name.	---	optional	optional	optional
Input Type for the input parameter field: ■ `Single Line Text` ■ `Multi Line Text` ■ `Enumeration` = a short list of valid values for this input parameter. Enter the valid values in **Possible Values**, separating each with a comma. ■ `Number` = numbers only, along with decimals and commas. ■ `Boolean` = true or false as the only valid values.	---	---	optional	optional

Property	Hidden	Read-Only	Indirect	Direct
■ `DateTime` = dates, times or both.				
Validation for the input parameter field: ■ `Any character` = no validation is performed. This is the default. ■ `Only Alphanumeric` = alphabetical characters and numbers only. ■ `Only Alphabets` = alphabetical characters only, no numbers. ■ `Only Numbers` = numbers only, no alphabetical characters.	---	---	optional	optional

Select and Arrange Views

Clear or set the options for each view you want to include in this app in the **Select View** step. You can also change the order of views which affects where they appear in the layout for the app. Simply drag selected views and drop them in the order that you want them.

Note: The views you choose also affect whether the app can be used in mobile devices. Mobile apps must have at least one view that is mobile compatible. See "About Desktop and Mobile View Compatibility" on page 903 for more information.

Preview and Finalize the Presentation

The **Preview** step lets you finalize the presentation for the app and preview your choices for desktop and mobile devices:

- Preview the App in Different Devices and Orientations
- Select the Layout for Views
- Select a Theme
- Set Pagination and Miscellaneous Properties
- Sorting and Filtering Properties

Preview the App in Different Devices and Orientations

Use the device toobar to change the preview for the app to:

Important: The previews for mobile devices may not include all views you have chosen for this app. Or a mobile preview can be completely blank if the app is not mobile compatible. See "About Desktop and Mobile Compatibility for Apps" on page 1111 for more information.

Select the Layout for Views

For apps with multiple views, you can also change the layout used to arrange each view in **Preview**:

	Tabbed = each view appears in a separate tab that users can click to switch views.
	Combo = the app has a single current view which users can switch by selecting a different view from a pull-down list.
	Two Column = views are organized in a two-column grid.
	Stacked = views are stacked vertically in one column.
	Wide Top Row = similar to a two-column layout, except the first row spans both columns.

If needed, return to **Select Views** and change the order of the views to get the desired effect

Select a Theme

App styling

Themes, in the **Preview** step, choose a color for the window, border and title bar for apps. If you have multiple views and choose to show the view titles, this also sets the color for the view title bar. The theme defaults to gray, simply choose another color if desired.

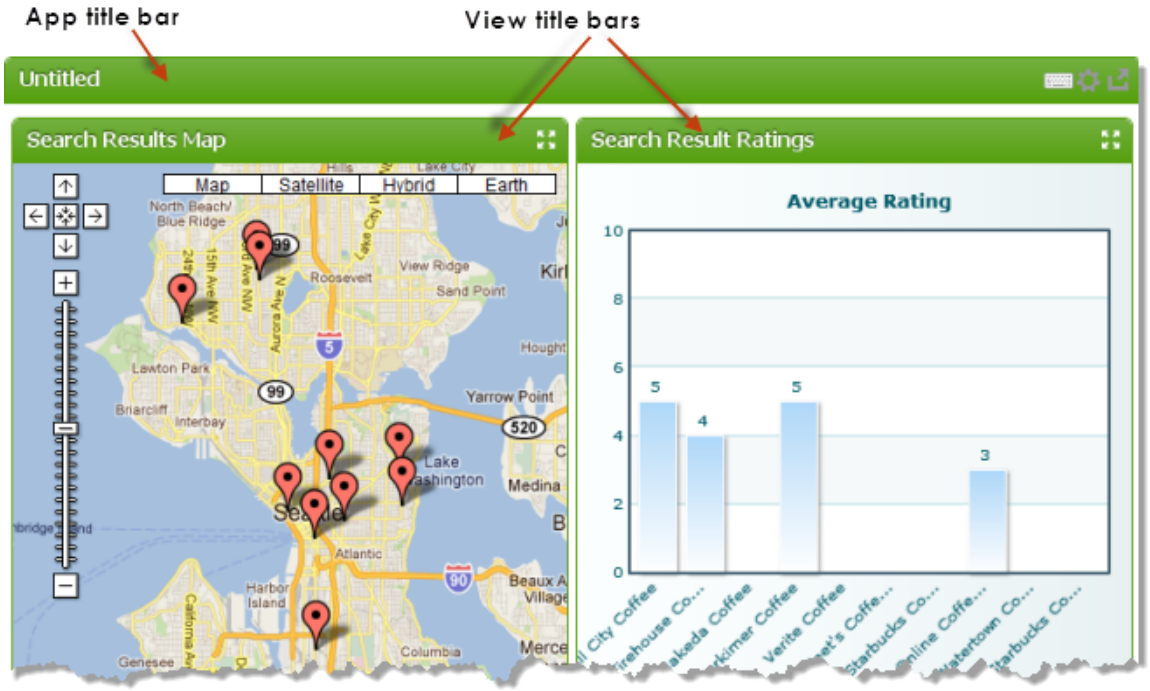


Chart styling (default themes and custom themes)

Choose a **Charts Theme** in the drop-down menu. Charts themes are predefined color sets for the look and feel of your charts (default themes are None, Carbon, Fint, Ocean, Zune). The selected theme will be applied to all displayed charts in your app.

In addition you can add your own custom themes to the **Chart Theme** drop-down menu. Therefor insert your chart theme file in the following folder and use the code block below.

MashZoneNG-install/ apache-tomee-jaxrs/webapps/presto/hub/third-party/ FusionChartsXT/themes/v0.0.3/fusioncharts.theme.custom.js

```
Presto.FusionCharts.ThemeManager.add([
  {
     name:"theme-name",
      theme:{
       base:{
                    chart:{}
                    }
                  }
            //here the user must be follow the sample theme there.
            }
            ]);
```

The added chart theme option will only applied to this specific app. If you want to set a global default chart theme, then you must set this option in Admin Console -> Platform features -> Look and feel -> Charts tab.

Set Pagination and Miscellaneous Properties

You also set *pagination* and miscellaneous properties for the app in **Preview**.

Many apps have repeating items in their information, sometimes only a few and sometimes a very large number of items. Showing all the items at once can be overwhelming to users and can make the app perform very slowly.

Pagination lets you control how many repeating items appear on one *page* (display of the app) and provides buttons to allow users to page through the data. However, pagination may not be appropriate for some kinds of views, such as a pie chart or some types of maps, where it is important to show all items together.

The two properties that you can set for pagination in the **Preview** step include:

- *Enable pagination*: this is set by default which adds a pagination toolbar in *every* view of the app to allow users to flip to other pages. Clear this option to show all repeating items in every view of the app.

  Note: Because pagination for the app applies to *all* of views in the app, it is sometimes better to create several basic apps with individual views so that you can control pagination individually. If you want these views to appear together, use a workspace to group the individual apps.

- *Rows per page*: this determines how many repeating items can appear on one page within the app. This defaults to 20, but you can choose another value.

Miscellaneous properties for the app include:

- **Show View Title** is set by default, which adds a title bar above each view along with the **Maximize V iew** toolbar button. Clear this option to remove view title bars everywhere this app is published.

  MashZone NextGen developers have additional ways to control title bars and toolbar buttons for apps in different contexts. See "Title Bars and Toolbar Buttons in Apps" on page 1112 for more information.

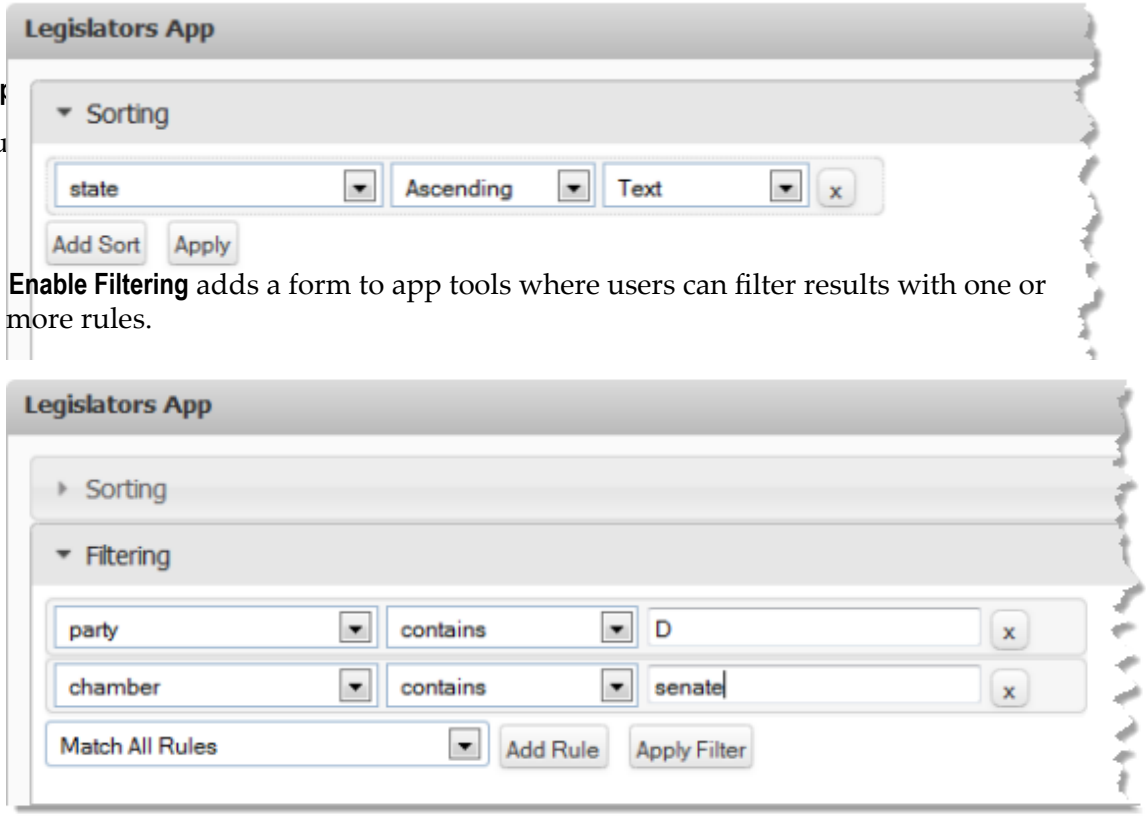
- **Height** and **Width** default to `auto` which allows the app to fit into the height or width of the page area wherever the app is published. You can set a suggested maximum height or maximum width as a number of pixels, although this is less flexible and is not guaranteed to be used.

  Note: Changing the width and height does *not* change the preview of the app, but may change the size of the app when it is published or embedded in other environments.

- **Auto Refresh Interval** defaults to zero. Users must manually refresh data with the Refresh toolbar button on the app's title bar. If the data for this app changes frequently, set this to the number of seconds that should elapse before the app automatically refreshes its data.

Sorting and Filtering Properties

You can add tools for other users to allow them to dynamically sort or filter results for this app based on their own criteria. These tools appear, along with the form for input parameters, when users click the



App

You

-
- **Enable Filtering** adds a form to app tools where users can filter results with one or more rules.

Save the App

Once you are satisfied with app, continue to the **Done** step to define which devices the app can be used in and to save this app.

1. Enter a **Name** for the app.

   This name also typically appears as the title for the app. You can hide the title, if you choose, when you use it in Mashboard or the AppDepot or when you publish or embed the app in other environments.

   MashZone NextGen uses the app name to assign a unique identifier to the app. App names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: _ ~ - * ' .

2. Enter a short **Description** for the app.
3. Optionally, select or enter meta data for this app including:
   - **Category**= the primary purpose, product, area of interest, aspect or other grouping for this artifact or resource. Categories define what an artifact pertains to. Categories are defined byMashZone NextGenadministrators.
   - **Provider**= the organization, department or group who provides or is responsible for the information in this artifact or resource. Providers define who or where information comes from. This can be external sources or systems or

groups within your own organization. Providers are defined byMashZone NextGenadministrators.

- **Tags**= one subject, purpose or other aspect of this artifact or resource. Tags define a finer grain of what an artifact is about. Artifacts can have any number of tags. Tags are defined by users.

  You can enter multiple tags, separate by commas. If the tag does not already exist, this also creates a new tag.

4. Select which devices the app can be used in. The **Desktop** option is set, by default. If this app is mobile-compatible, set the **Phone** and/or **Tablet** options as desired. See "About Desktop and Mobile Compatibility for Apps" on page 1111 for more information.
5. Click **Create this App**.

The app is turned on and is now visible in MashZone NextGen Hub. You can:

- Open the app's artifact page to preview the app, update app information, grant permission to other users to work with the app, share the app and manage the app. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 and "Common Tasks for Mashables, Mashups and Apps" on page 85 for more information.
- Publish the app to the AppDepot to make it available to other users in AppDepot (for desktop devices) or in the MashZone NextGen Mobile apps (for mobile devices). See "Publish Apps to the AppDepot" on page 1114 for details.
- Embed or publish the app to web pages or SharePoint. See "Publishing, Managing, Sharing and Using Apps" on page 1101 for more information.
- Use this app in a workspace in Mashboard. See "Create Workspace Apps with Mashboard" on page 1116 for more information.

Publishing, Managing, Sharing and Using Apps

The app artifact page lets you preview that app. Use the menus in this page to edit, manage, publish and perform other tasks with the app. You can also use the toolbar to share the app directly with another user or mark it as one of your favorites.

Note: Some actions in an app artifact page may not be accessible based on your MashZone NextGen permissions.

- Use the **Publish** menu to:
  - Publish Apps to the AppDepot
  - Publish an App in SharePoint
  - Embed an App in HTML Pages

- Use the **Edit** menu to Edit App Properties. You can also open the app in the App Editor from this menu.
- Use the **Show** menu to view or update basic information, view the audit log or find dependencies (related artifacts) for the app. See "Common Tasks for Mashables, Mashups and Apps" on page 85 for more information.

  Developers can also Edit and Test an Open App to edit the code and resources for an app.
- Use **Manage** to:
  - Grant or revoke permissions to run the app. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.
  - Add screenshots for the app. These are useful for users in AppDepot to help them determine if an app may be useful to them.

    Take one or more screens shots of your app. Click **Manage > Screenshots** and upload your images.
  - Change the app's status, remove the app or feature this app. See "Common Tasks for Mashables, Mashups and Apps" on page 85 for more information.
- You can also change the thumbnail for the app from the thumbnail on the artifact page. See "Add a Thumbnail" on page 89 for instructions.

Embed an App in HTML Pages

To embed an app in an HTML page in your intranet or in a web application, you must update the HTML code for that page. You may need to edit the page directly, in a text editor or your choice, or use features in the system that manages the page.

To embed an app

1. In MashZone NextGen Hub, open the app from search results, favorites or bookmarks.
2. If desired, change the input parameters and other visual settings that you want to use for this app in this HTML page. See "Edit App Properties" on page 1104 for instructions.
3. Click **Publish > Embed**.
4. Copy the <script> tag from the Embed window and paste this tag into the HTML page where you want to embed this app.

Publish an App in SharePoint

You can publish an app to a specific site and Web Part page in SharePoint. This option is only available if your MashZone NextGen administrator has installed and configured the MashZone NextGen Add-On for SharePoint Add-On.

To publish an app

1. In MashZone NextGen Hub, open the app from search results, favorites or bookmarks.
2. If desired, change the input parameters and other visual settings that you want to use for this app in SharePoint. See "Edit App Properties" on page 1104 for instructions.
3. Click **Publish >** **SharePoint**.
4. Click **Publish to SharePoint**.
5. Choose the **SharePoint Connection** where you want to publish this app.

   Note: If no SharePoint connections have been configured in MashZone NextGen, this field has an error message. You cannot publish apps to SharePoint until a connection has been configured by a MashZone NextGen administrator.

   If the SharePoint connection you need to work with is not in this list, contact your MashZone NextGen administrator.

6. If needed, update the **User Credentials** to use to publish this app in SharePoint and click **Update**.

   Important: Updating these credentials creates or updates *your* MashZone NextGen User Attributes for this SharePoint server. This can affect how other mashups or mashables that work with SharePoint behave for you if they are configured to use the current user's credentials.

   The permissions for the credentials you use determine which SharePoint sites and Web Part pages you may publish the app in. Based on configuration:

   - This may default to the credentials defined for the connection you have selected, also known as the *service account*.
   - Or you may be required to supply your credentials for this SharePoint connection.
   - Or the connection may disable the use of user credentials based on configuration and not allow updates.

   To remove your credentials and return these to the default credentials for this SharePoint server, click **Remove**. This also *removes* your MashZone NextGen User Attributes for this SharePoint server.

7. Choose the **Document Library** where you want to publish this app.
8. Choose the **Page Name** for the Web Part Page where you want to publish this app.
9. Choose the **Zone Name** that this app should fill.
10. Click **Publish**.

Edit App Properties

To edit app properties

1. Open the artifact page for the app from search results, favorites or other links.
2. Click **Edit >** **App Properties**.
3. Edit the properties provided. For basic apps, you use the App Maker wizard to update:
   - "Input Parameters" on page 1104 in **Configure Inputs**.
   - Views and view order in **Select Views**.

     Simply clear or set options for the views to use in the app.

     Drag views in the list to change the order they appear in the app.
   - Layouts, Themes, Pagination and Miscellaneous Properties and Sorting and Filtering Properties properties in the **Preview** step.

     You can also preview all of your changes for desktop and mobile devices using the preview device toolbar:

   - Device Compatibility and General Properties in **Done**.

   Use the Preview tab to review your changes.

   For custom or workspace apps, see "Edit Properties for Custom or Workspace Apps" on page 1110 for details.

Input Parameters

If the app has input parameters, you can update how these parameters are handled in the **Configure Inputs** step.

For desktop apps *only*, you can choose one of two methods to open the form where users set input parameters:

- **Dialog** opens the form as a window on top of the app when it first opens. Users can reopen this form to change inputs using the **App Tools** button in the app's title bar.

> **Note:** Mobile apps always place the input form in a separate window and provide a button in the title bar to open this window.

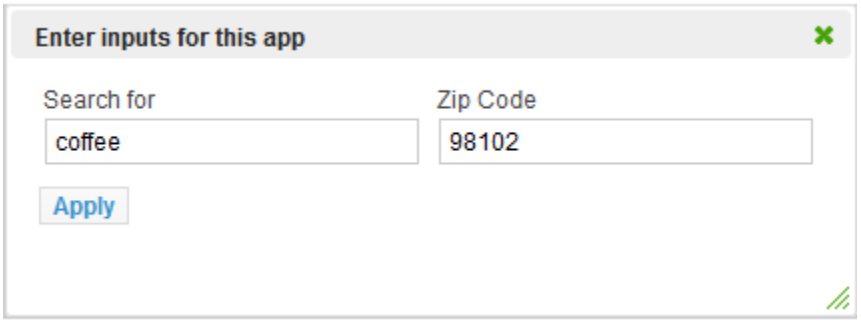
- **View** makes the form one view that is always open within the layout of the app.

Select a parameter to update how that parameter is handled. There are four patterns for input parameters:

- *Hidden*: for parameters that users should *not* change and do not need to see, such as technical keys necessary to run a mashable. Hidden properties are visible for wiring apps in Mashboard, however.
- *Read-only*: for parameters that users should *not* change but do need to see. Read-only input parameters are visible in app settings in Mashboard, in apps users have saved to Favorites in the AppDepot, and for wiring apps in Mashboard.
- *Indirect*: for parameters whose values are supplied by saved preferences for users in the AppDepot.

  Indirect input parameters are not present in the input form for an app. If all input parameters for an app are indirect, the app has no input form.
- *Direct*: for input parameters that either users supply directly in the input form, at any time when they use this app, or are supplied automatically through wiring in a workspace app.

  Direct input parameters appear in the input form for the app, unless that form has been suppressed in a workspace app. (See "Hiding Input Forms for Apps in a Workspace" on page 1181 for more information.)



  Direct input parameters give users control. Users cannot, however, override direct input parameters with their saved preferences for a custom Favorite App in the AppDepot.

Property	Hidden	Read-Only	Indirect	Direct
Editable	no	no	yes	yes
Visible	no	yes	no	yes

Property	Hidden	Read-Only	Indirect	Direct
Required	---	---	optional	optional
Default Value	required	required	optional	optional
Label defaults to the input parameter name.	---	optional	optional	optional
Input Type for the input parameter field: ■ `Single Line Text` ■ `Multi Line Text` ■ `Enumeration` = a short list of valid values for this input parameter. Enter the valid values in **Possible Values**, separating each with a comma. ■ `Number` = numbers only, along with decimals and commas. ■ `Boolean` = true or false as the only valid values. ■ `DateTime` = dates, times or both.	---	---	optional	optional
Validation for the input parameter field: ■ `Any character` = no validation is performed. This is the default. ■ `Only Alphanumeric` = alphabetical characters and numbers only. ■ `Only Alphabets` = alphabetical characters only, no numbers. ■ `Only Numbers` = numbers only, no alphabetical characters.	---	---	optional	optional

Layouts

For apps with multiple views, you can also change the layout used to arrange each view in **Preview**:

	Tabbed = each view appears in a separate tab that users can click to switch views.
	Combo = the app has a single current view which users can switch by selecting a different view from a pull-down list.
	Two Column = views are organized in a two-column grid.
	Stacked = views are stacked vertically in one column.
	Wide Top Row = similar to a two-column layout, except the first row spans both columns.

Themes

App styling

Themes, in the **Preview** step, choose a color for the window, border and title bar for apps. If you have multiple views and choose to show the view titles, this also sets the color for the view title bar. The theme defaults to gray, simply choose another color if desired.

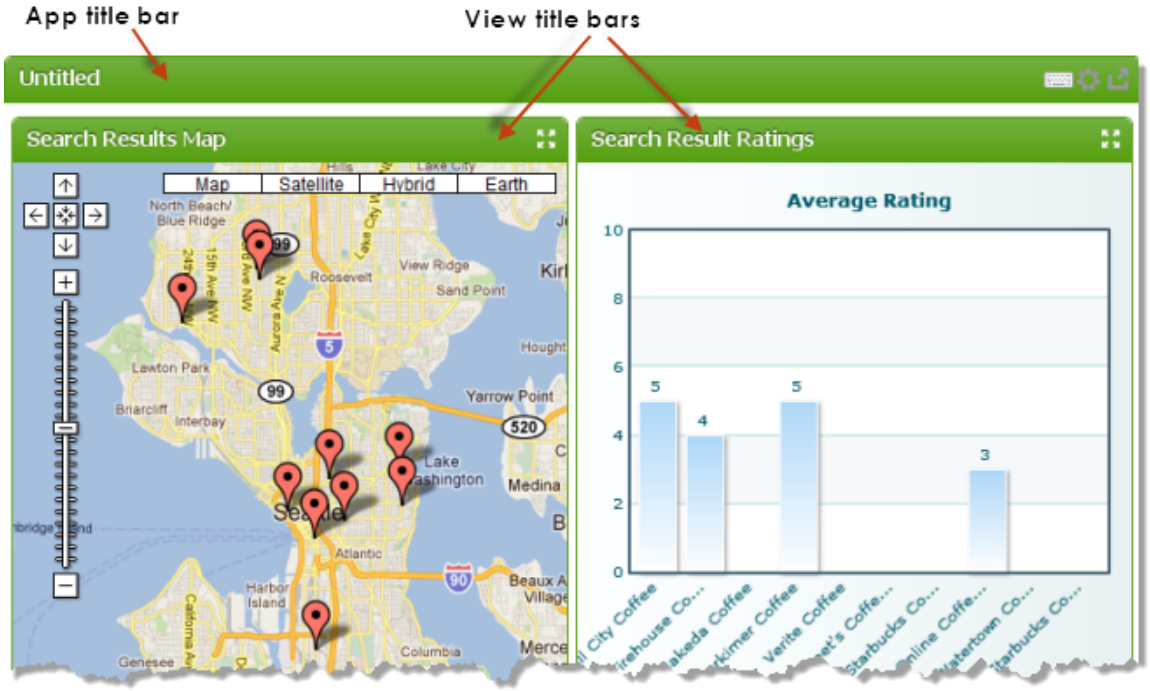


Chart styling (default themes and custom themes)

Choose a **Charts Theme** in the drop-down menu. Charts themes are predefined color sets for the look and feel of your charts (default themes are None, Carbon, Fint, Ocean, Zune). The selected theme will be applied to all displayed charts in your app.

In addition you can add your own custom themes to the **Chart Theme** drop-down menu. Therefor insert your chart theme file in the following folder and use the code block below.

MashZoneNG-install/ apache-tomee-jaxrs/webapps/presto/hub/third-party/FusionChartsXT/themes/v0.0.3/fusioncharts.theme.custom.js

```
Presto.FusionCharts.ThemeManager.add([
  {
     name:"theme-name",
      theme:{
       base:{
                   chart:{}
                   }
                  }
           //here the user must be follow the sample theme there.
           }
           ]);
```

The added chart theme option will only applied to this specific app. If you want to set a global default chart theme, then you must set this option in Admin Console -> Platform features -> Look and feel -> Charts tab.

Pagination and Miscellaneous Properties

Many apps have repeating items in their information, sometimes only a few and sometimes a very large number of items. Showing all the items at once can be overwhelming to users and can make the app perform very slowly.

Pagination lets you control how many repeating items appear on one *page* (display of the app) and provides buttons to allow users to page through the data. However, pagination may not be appropriate for some kinds of views, such as a pie chart or some types of maps, where it is important to show all items together.

The two properties that you can set for pagination in the **Preview** step include:

- *Enable pagination*: this is set by default which adds a pagination toolbar in *every* view of the app to allow users to flip to other pages. Clear this option to show all repeating items in every view of the app.

  Note: Because pagination for the app applies to *all* of views in the app, it is sometimes better to create several basic apps with individual views so that you can control pagination individually. If you want these views to appear together, use a workspace to group the individual apps.

- *Rows per page*: this determines how many repeating items can appear on one page within the app. This defaults to 20, but you can choose another value.

Miscellaneous properties for the app include:

- **Show View Title** is set by default, which adds a title bar above each view along with the **Maximize V iew** toolbar button. Clear this option to remove view title bars everywhere this app is published.

  MashZone NextGen developers have additional ways to control title bars and toolbar buttons for apps in different contexts. See "Title Bars and Toolbar Buttons in Apps" on page 1112 for more information.

- **Height** and **Width** default to `auto` which allows the app to fit into the height or width of the page area wherever the app is published. You can set a suggested maximum height or maximum width as a number of pixels, although this is less flexible and is not guaranteed to be used.

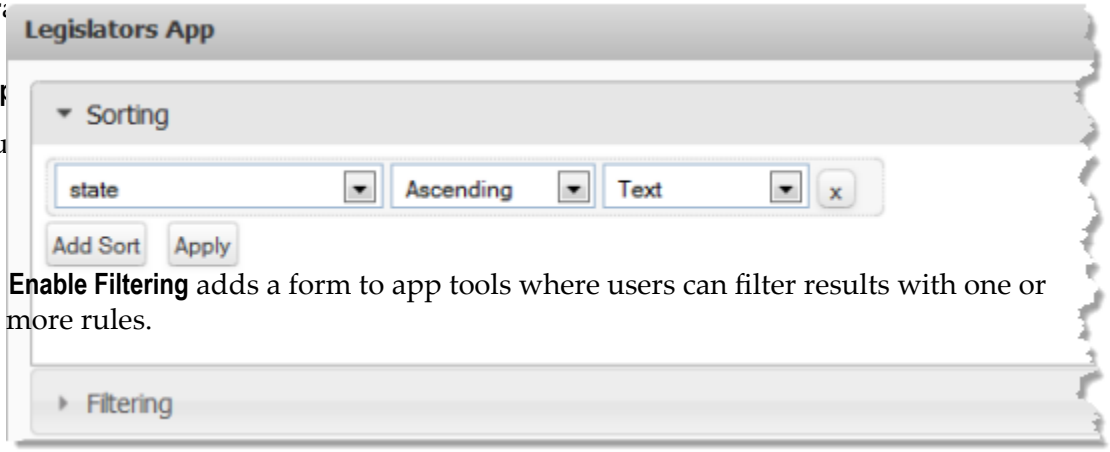
  Note: Changing the width and height does *not* change the preview of the app, but may change the size of the app when it is published or embedded in other environments.

- **Auto Refresh Interval** defaults to zero. Users must manually refresh data with the Refresh toolbar button on the app's title bar. If the data for this app changes frequently, set this to the number of seconds that should elapse before the app automatically refreshes its data.

Sorting and Filtering Properties

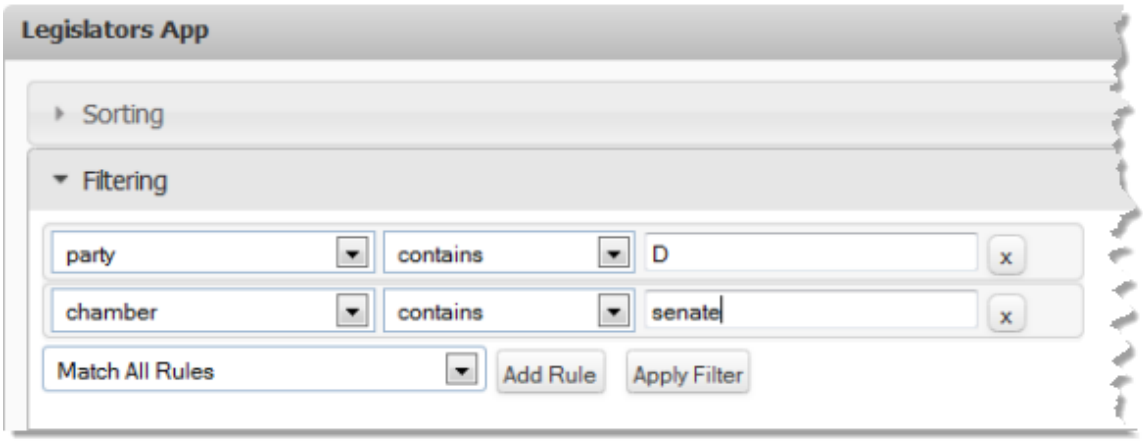
You can add tools for other users to allow them to dynamically sort or filter results for this app based on their own criteria. These tools appear, along with the form for input para

App



You

-
- **Enable Filtering** adds a form to app tools where users can filter results with one or more rules.



Device Compatibility and General Properties

In the **Done** step, you can edit the **Name**, **Description** and other meta data for the app.

You can also change which devices the app can be used in. Clear or set the **Desktop**, **Phone** and/or **Tablet** options as desired. See "About Desktop and Mobile Compatibility for Apps" on page 1111 for more information.

Edit Properties for Custom or Workspace Apps

The custom app or workspace app properties you can update include:

- **Title** = the app's title which appears in the title bar, if that is enabled. For workspaces, this is the title that appears in the workspace tab in Mashboard.
- **Height** = the height in pixels to display this app. This defaults to `auto` which allows the app to fit into the size of the page area when it is used. You can change this to a number of pixels to use as a suggested maximum height.
- **Width** = the width in pixels to display this app. This defaults to `auto` which allows the app to fit into the size of the page area when it is used. You can change this to a number of pixels to use as a suggested maximum.
- **Refresh Interval** = this defaults to 0, which means that users must manually refresh the app. Set this to the number of seconds that should elapse before the app automatically refreshes its data. Setting a refresh interval is useful for apps that work with data which changes frequently.
- **Hide the App Title Bar** = set this option if you do not want the app's title or toolbar buttons to appear.
- *input-parameter* = if the app has input parameters, options appear for each parameter so that you can update the default value.

About Desktop and Mobile Compatibility for Apps

Most apps are designed, by default, to be desktop compatible. They run properly in browsers on laptops or other personal computers. Their view is appropriate and responds appropriately to mouse actions such as click or scroll.

Mobile compatibility means that the app has been optimized to function in mobile devices on the mobile operating systems that MashZone NextGen supports. MashZone NextGen built-in views that are mobile compatible also recognize and respond, where appropriate, to common gestures such as tap.

The format of the view in a mobile device may be different than its desktop counterpart. For basic apps with multiple views, the layout used in mobile devices may differ from the layout used in desktop devices.

In workspace apps with compatible layouts, mobile and desktop layouts remain the same. Basic apps or built-in MashZone NextGen views included in the workspace, however, are shown with their desktop view.

Note: Currently, MashZone NextGen supports mobile apps for iOS (iPhone or iPad) devices.

Apps can be both desktop- and mobile-compatible. Or they can be designed for one specific device. Device compatibility for an app depends on how the app was created:

Type of app	Desktop Compatible	Mobile Compatible
Basic apps created in the App Maker wizard	Yes, if: ■ At least one view is desktop compatible and ■ The app was flagged for desktop devices when it was created All of the views bundled in MashZone NextGen are desktop compatible. Consult your MashZone NextGen administrator for information on any pluggable views that have been added to the MashZone NextGen View Gallery.	Yes if: ■ At least one view is mobile compatible and ■ The app was flagged for that type of mobile device (phone or tablet) when it was published to the AppDepot. See "About Desktop and Mobile View Compatibility" on page 903 and "Preview and Finalize the Presentation" on page 1096 for more information.

Type of app	Desktop Compatible	Mobile Compatible
Workspace apps created in Mashboard	Yes, if at least one view or app in the workspace app is desktop compatible.	Yes, if: ■ The workspace layout is a column or table layout and ■ The workspace was flagged for that type of mobile device (phone or tablet) when it was published to the AppDepot. Workspace apps that include apps or views that are not mobile compatible do attempt to render these components using their desktop view. In some cases however, the result may be unacceptable.
Custom apps written by MashZone NextGen developers	Yes if the app: ■ Has been designed for desktop browsers and ■ Is flagged in its App Spec as desktop compatible See "App Layout" on page 1306 for more information.	Yes if the app: ■ Has been designed as HTLM5 compatible, ■ Uses a mobile JavaScript library that supports gestures, such as jQuery Mobile and ■ Is flagged in its App Spec as compatible for that type of mobile device (phone or tablet) See "App Layout" on page 1306 for more information.

Title Bars and Toolbar Buttons in Apps

MashZone NextGen basic and custom apps have a title bar, by default, and basic apps may also optionally have a title bar for each view in the app. Title bars include buttons to provide common functionality to any user who has permission to use the app. See Buttons in the Default Title Bars for details.

These title bars and buttons are generally visible within MashZone NextGen. Workspace apps override the default title bars and toolbar buttons but do provide most of the same functionality. Other destinations, however, *may hide* these default title bars and buttons.

Each destination may allow users to re-enable the title bars or buttons or provide similar functionality somewhere within its own user interface. When apps are published to SharePoint, for example, title bars are hidden by default to optimize space on Web Part pages. Users who can edit these Web Part pages, however, can make these title bars visible if desired.

MashZone NextGen developers can also control title bar and toolbar button visibility, if needed, when apps are published to different destinations. See "<presto-meta>" on page 1306 and "Parameters for App URLs" on page 1294 for details on the flags and parameters that affect title bars and toolbar buttons for apps.

Buttons in the Default Title Bars

The default app and view title bars include the following buttons to for users:

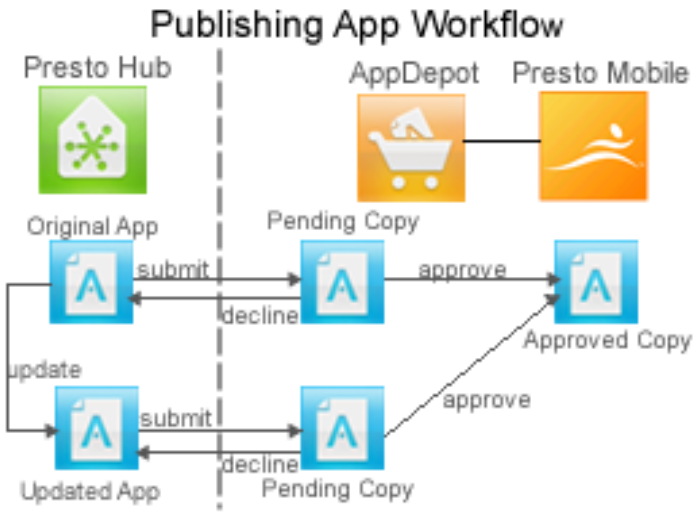
Button	Description	Title Bar	
		App	View
Tools	Opens the forms that are defined for this app to: ■ Update the input parameters to see different information, if this app accepts parameters. ■ Add or change sorting for the app, if the app is configured for dynamic sorting. You choose wether to enable this button when you create the app.	✓	

Button	Description	Title Bar	
		App	View
	■ Add or change filtering criteria for the app, if the app is configured for dynamic filtering. You choose wether to enable this button when you create the app.		
Refresh	■ For apps that do *not* have automatic refreshes enabled, this manually refreshes the data for this app. ■ For apps that do have automatic refreshes enabled, this button is a toggle to turn off/turn on automatic refreshes.	✓	
Open in New Window	Opens the app by itself in a new window or browser tab.	✓	
Share Menu	Options to: ■ Share a link to the app with other users. ■ Get the embed code for this app. ■ Publish the app to SharePoint.	✓	
Maximize View	Expands this view to fit the entire width of the app or collapses the view back into the app's layout.		✓

Publish Apps to the AppDepot

You must publish apps to make them available in the AppDepot or the MashZone NextGen Mobile apps where other users can work with them. Users find and use desktop apps in the AppDepot and find and work with mobile apps in the MashZone NextGen Mobile apps.

The publishing process manages how apps are made available to users and also manages updates to help ensure minimal disruptions. Publishing creates a 'live' copy of the app that is separate from your app in the MashZone NextGen Hub.



To publish apps

1. Provide meta data and complete any other app requirements defined by your organization.

   Meta data, such as screen shots and information about the app serves as *provenance* to help users discover apps in the AppDepot or the MashZone NextGen Mobile apps and determine which apps they want to use. Your organization determines what meta data is required plus any other standards that apps should meet before they are published to the AppDepot and the MashZone NextGen Mobile apps.

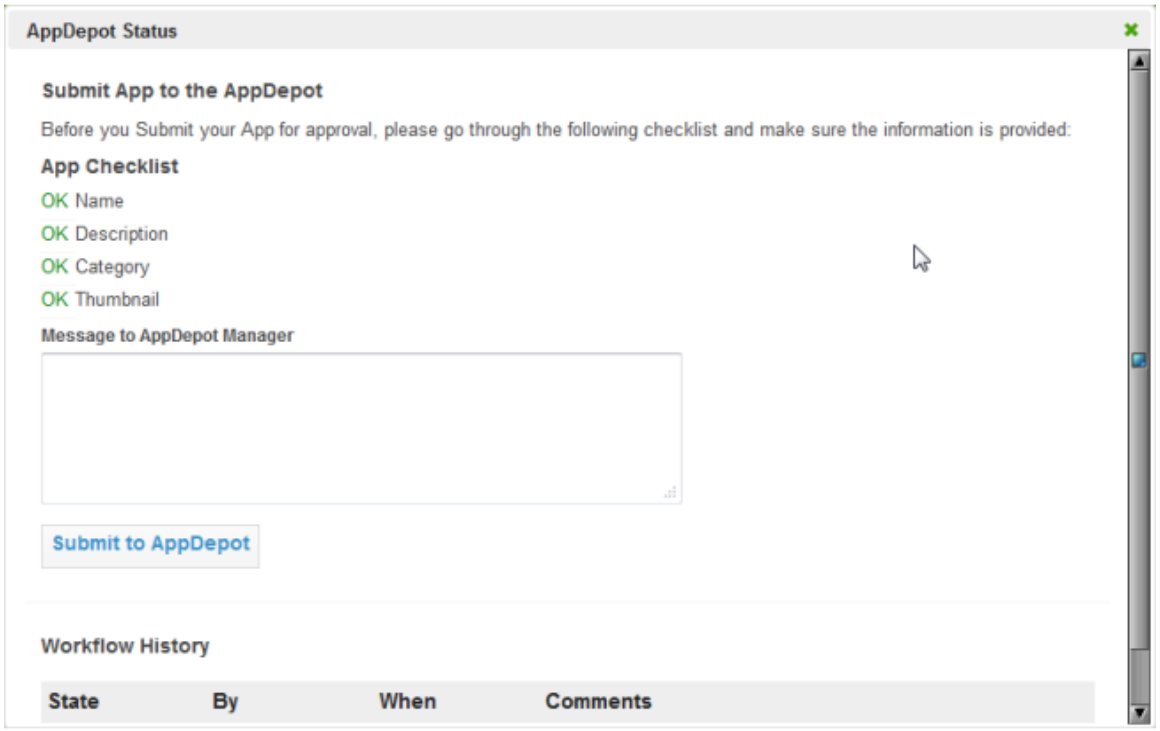
   Common tasks include:

   - Filling in a description, category, provider or tags to help users find and understand the app. See "Add or Update Meta Data: Description, Category, Provider and Tags" on page 87 for instructions.
   - Adding a custom thumbnail image. See "Add a Thumbnail" on page 89 for instructions.
   - Add screenshots of the app to help illustrate the different views or uses of the app.

2. Submit the app to the AppDepot.

   a. In MashZone NextGen Hub, open the app artifact page from search, your favorites or other links.

   b. Click **Publish > To AppDepot**.



c. Enter a message to the AppDepot manager and click **Submit to AppDepot**.

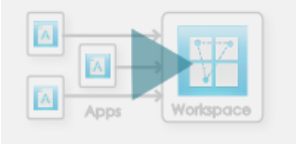
   The status changes to pending and a new log entry appears for the Workflow History. You should also see a notice that the app has been submitted in the Notifications section on the MashZone NextGen Hub home page.

3. An AppDepot Manager must then review and approve or decline the app. MashZone NextGen administrators should see "Managing Pending Apps in the AppDepot" on page 1187.

   Once your app is approved, it appears in **All Apps** or in search results in either or both:

   - The AppDepot, if your app is compatible with desktop browsers.
   - One or both of the MashZone NextGen Mobile apps on mobile phones or mobile tablets, if it is compatible with that type of mobile device.

Create Workspace Apps with Mashboard



Workspaces group individual apps in a useful layout that you save as a new app. Workspaces can also:

- Contain views of mashables or mashups.

  Note: When views are added directly to a workspace, Mashboard automatically creates a basic app containing just that view. This app is available only within the workspace.

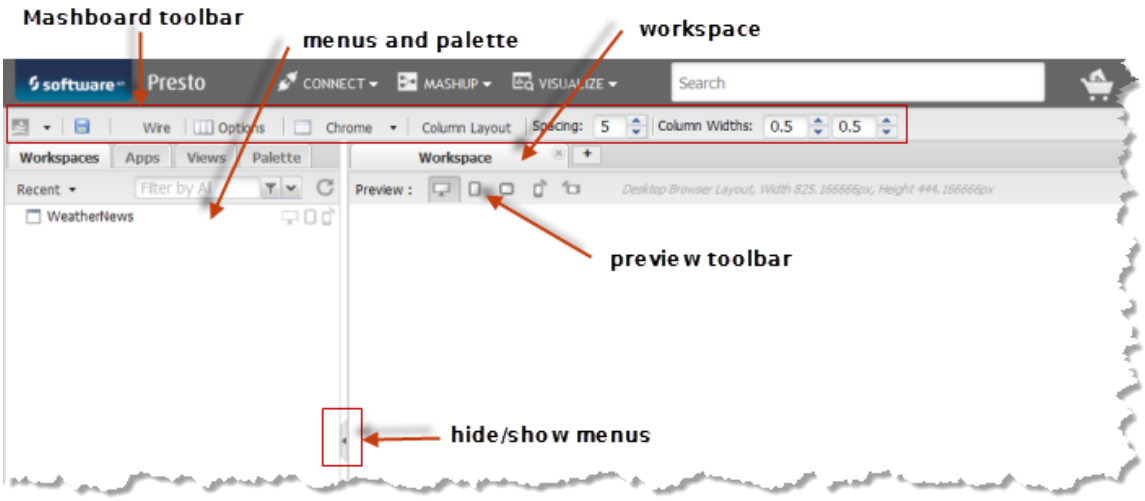
  Thus the word *app* or *basic app* in a workspace may refer to existing apps created independently or to views added directly to a workspace.

- Contain input forms for parameters or forms to filter app content.
- Contain other types of content such as web pages, videos and other media.
- Wire interactions between the apps that they contain to synchronize their data or behavior. Or wire interations to input or filter forms to give users options when they use workspaces.

  Wiring interactions ties apps, and optionally the controls in forms, into a composition that works together. Wiring is also sometimes called inter-app communication.

Workspaces most commonly contain apps or other content that focus on one subject, such as a dashboard, or are designed for the use of a specific team. To create a workspace:

1. Select **Visualize >** Mashboard in the MashZone NextGen Hub main menu.



   See "Working in Mashboard" on page 1121 for more information on Mashboard menus, buttons and toolbars.

2. Click + to open a tab for a **New Workspace**, if needed, and choose a layout from the Layout Gallery:

Note: Some workspace layouts are compatible with mobile devices and some are only for desktop browsers. The Layout Gallery shows device compatibility for each layout with these icons:

(desktop) (mobile phone) (mobile tablet)

- `Column Layouts`: apps appear side-by-side in one row. You can add multiple apps to a column which will stack vertically.

  This defaults to two columns but you can adjust the layout. See the next step for links to adjustment techniques.

- `Table Layout`: arrange up to 64 apps in a variety of grids.

  This layout default to two columns and two rows, but you can adjust the layout. See the next step for links to adjustment techniques.

- `Desktop Layout`: users can move apps around in this layout to any position within the single 'page' of the workspace. Users can also minimize apps to a *dock* or task bar and maximize apps from the dock, or use a context menu to tile or cascade apps.

- `Tabbed Layout`: the workspace can contain multiple tabs, each of which has its own layout, containing one or more apps. Individual tab layouts can be `Column`, `Table` or `Desktop` or tabs can contain a `Web Page`. The tabs provide direct navigation to their contents.

- `Paged Layout`: like a tabbed layout, this has multiple 'pages' but no tabs. Each page has its own layout, containing one or more apps. Individual page layouts can be `Column`, `Table` or `Desktop` or pages can contain a `Web Page`. Users navigate to different pages using a paging toolbar.

  See also "Add a Drill Down App in a Workspace" on page 1158 for another form of paged navigation through apps.

3. Add content to the workspace and adjust the layout using these techniques:

   - Drag views from the **Views** menu or drag apps from the **Apps** menu and drop them in the desired location in the workspace.

See "Find Views, Apps or Workspaces in Mashboard" on page 1123 for tips on finding views and apps of interest. See also "Views Added Directly to a Workspace" on page 1124.

- Add chart from PPM to the workspace. See "Add a PPM Chart to a Workspace" on page 1147 for information on using this custom app in a workspace.
- Add real-time line chart for events. Real-time charts are designed to handle streaming updates for events. See "Add a Real-Time Line Chart" on page 1142 for instructions.
- To include drill downs, input or filter forms, or other types of content in the workspace, use the **Palette** menu.

  Note: the term *gadget* refers to any or all of the different types of inputs, other content or widgets that you can add to a workspace.

  See the following topics for instructions:

  - "Add a Drill Down App in a Workspace" on page 1158
  - "Add an Input Form for All Apps In a Workspace" on page 1164
  - "Add a Filter for Apps in a Workspace" on page 1161
  - "Add Google Gadgets to a Workspace" on page 1170
  - "Add an HTML Widget to a Workspace" on page 1171
  - "Add Other Media as Objects to a Workspace" on page 1172
  - "Add Web Pages to a Workspace" on page 1173
  - "Add Gadgets With JavaScript to a Workspace" on page 1174
- Adjust the layout of the workspace. The layout options you have are different for each type of layout. See these topics for instructions:

Any Workspace	Column or Desktop	Table	Tabbed or Paged
"Copy a Workspace" on page 1128	"Add or Delete Rows and Columns to a Layout" on page 1131		"Add Tabs or Pages" on page 1140
"Move, Cut, Copy, Paste or Remove Apps or Views in Workspaces" on page 1127	"Initial Widths and Heights for Column or Table Layouts" on page 1133		"Delete Tabs or Pages" on page 1140

Any Workspace	Column or Desktop	Table	Tabbed or Paged
"Control Title Bars and Borders" on page 1128	"Control Padding for Column or Table Layouts" on page 1134		"Change Tab or Page Names" on page 1140
"Set Input Parameters, Pagination, Isolation or Auto-refresh" on page 1130	"Resize Apps in Column or Desktop Layouts" on page 1134	"Change Cell Width or Height with Spanning in Table Layouts" on page 1137	"Change the Order of Pages" on page 1140
"Set Data Share or Sort and Filter Tools for Views" on page 1131	"Set Desktop Layout Colors and Properties" on page 1141	"Fixed Row Heights for Table Layouts" on page 1138	"Set Other Tab or Page Configuration" on page 1140

4. Optionally, wire app interactions to synchronize their data or behavior. See "Wiring App Interactions in a Workspace" on page 1175 for instructions.
5. Once the workspace is laid out with the content you want, click **Save** and:
   a. Enter a **Name** for this workspace.

      This defaults to the name on the workspace tab in Mashboard, which typically is the name of the layout you chose for the workspace. Change this to a more meaningful name.

      MashZone NextGen uses the app name to assign a unique identifier to the app. App names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`
   b. Enter a **Title** for this workspace.

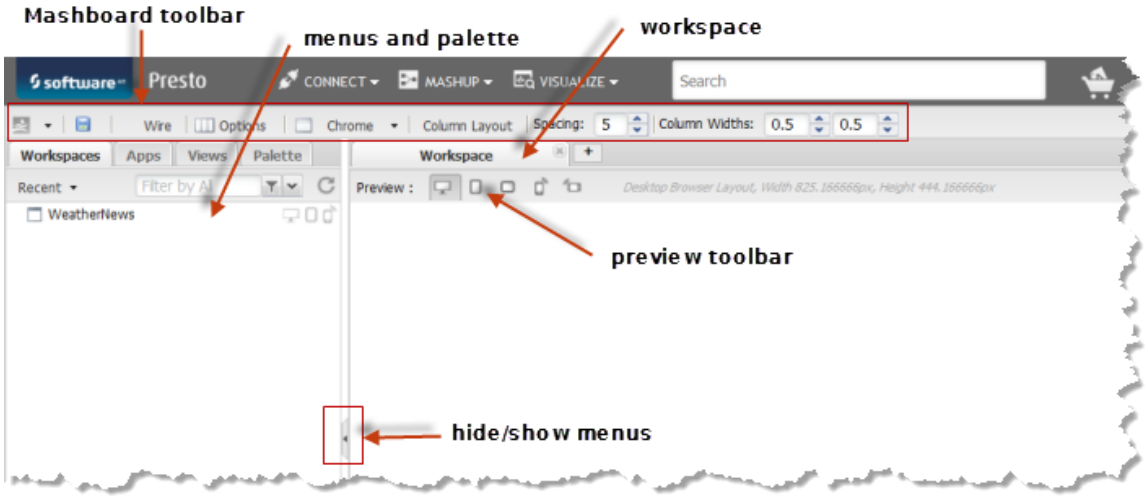
      This defaults to the workspace name, but you can change this as needed. Titles typically appear in the title bar for the app or workspace.
   c. Enter any of the optional meta data for the workspace. See "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 for more information.
   d. Choose the devices you want to make this workspace app available on:
      - `Desktop`
      - `Phone`
      - `Tablet`

e. Click **Create**.

6. The workspace is saved and is now visible in MashZone NextGen Hub as an app. It also now appears in the **Workspaces** tab in Mashboard.
7. Set run permissions for the workspace to enable other users to work with this app. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.

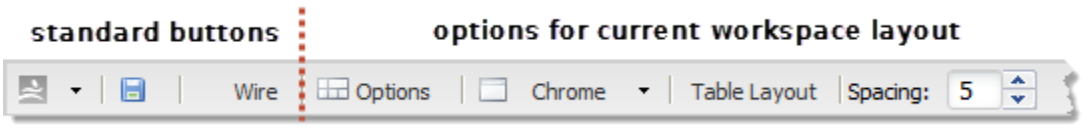
Working in Mashboard

Mashboard is a design tool to create workspace apps that combine views from different mashables or mashups, other existing apps and/or other gadgets into a single focus or work area. The menus, toolbars and other components in Mashboard are shown below:



- *MashboardToolbar*: with buttons to set options for Mashboard, to save a workspace or to wire workspace interactions. For more information, see
  - "Create Workspace Apps with Mashboard" on page 1116
  - "Wiring App Interactions in a Workspace" on page 1175

  This toolbar also contains buttons that are specific to the layout of the workspace that is currently open in Mashboard.



  For more information on layout-specific toolbar buttons, see the sections and links in "Edit Workspaces" on page 1126.

- *Menus and Palette*: to find workspaces, apps, views and other gadgets to use in workspaces as you create or edit them. This includes the:
  - **Workspaces** tab to find workspaces and edit them.
  - **Apps** tab to find apps and add them to a workspace.
  - **Views** tab to find views for mashables or mashups and add them to a workspace directly without having to create an app first.
  - **Palette** tab to add containers, forms, web pages or other gadgets to workspaces.

  For more information, see:

  - "Find Views, Apps or Workspaces in Mashboard" on page 1123
  - "Views Added Directly to a Workspace" on page 1124
  - "Add a PPM Chart to a Workspace" on page 1147
  - "Add a Real-Time Line Chart" on page 1142
  - "Add a Drill Down App in a Workspace" on page 1158
  - "Add a Filter for Apps in a Workspace" on page 1161
  - "Add an Input Form for All Apps In a Workspace" on page 1164
  - "Add Google Gadgets to a Workspace" on page 1170
  - "Add an HTML Widget to a Workspace" on page 1171
  - "Add Other Media as Objects to a Workspace" on page 1172
  - "Add Web Pages to a Workspace" on page 1173
  - "Add Gadgets With JavaScript to a Workspace" on page 1174

  You can also use the Hide/show buttons (below) in the right border of the menus:

  Hide allows you to expand workspaces across the full width of the browser. Show expands the menus once again.

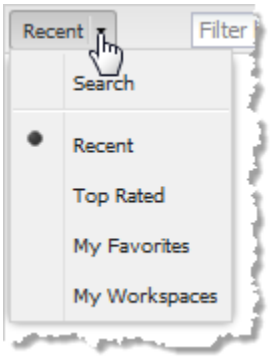
- *Workspace tabs* with one tab as the design area for each workspace that you have open. Each workspace tab also has a *preview toolbar* you can use to preview the workspace for desktop or mobile devices and different orientations:

Find Views, Apps or Workspaces in Mashboard

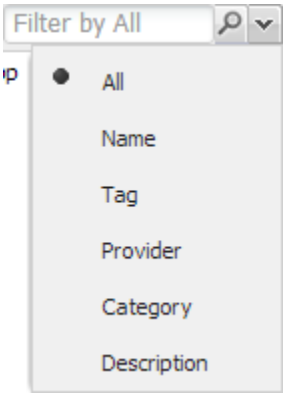
Mashboard provides easy access to workspaces, views and apps. The techniques you use to find workspaces, views and apps are similar:

- Open the **Workspaces**, **Apps**or **Views**tab.

  By default, these tabs list the most recently created workspaces, apps or mashables and mashups. Change the left menu to search or use other predefined filters.



- Select **Search**. By default, this searches for workspaces, apps or mashables and mashups by name, tag, category, provider and description. Then enter part or all of the phrase you want to search for.

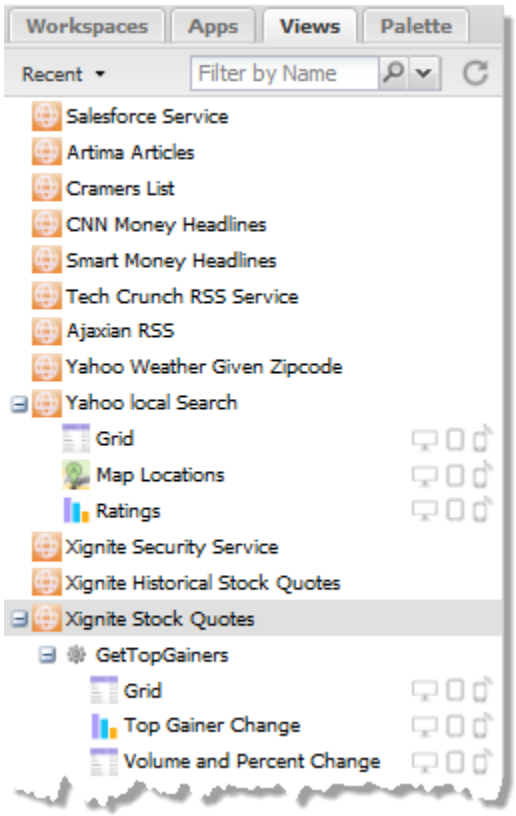


  You can also click to change the scope of the search to just name, tag, provider, category or description.

- Choose Top Rated or My Favorites to see workspaces, apps or mashables/mashups that are rated or that you own.

Finding Views

The Views tab lists mashables and mashups. To find a specific view, click the mashable or mashup.



If needed, select the operation you want to search for a view. Once you have the view you want, simply drag it into the workspace position where you want it to appear.

See also "Views Added Directly to a Workspace" on page 1124.

Views Added Directly to a Workspace

You can directly add views to a workspace from any mashable or mashup that you have permission to run, just as you can add apps. To add views directly, find the mashable or mashup and then find the view. See "Find Views, Apps or Workspaces in Mashboard" on page 1123 for techniques to find views. Then drag the view into the appropriate area in the workspaces.

Basic App Wrapper for Views

When you add a view directly to a workspace, Mashboard creates a basic app to wrap this view. This basic app is only visible in the workspace. It has the same properties as any basic app including:

- An input form if the mashable or mashup for this view has input parameters. By default, this form contains all input parameters.
- Properties to set the theme, pagination and so on.

- Forms to allow all users to sort and/or filter results data for the view can also be enabled. These forms are disabled by default.
- A Data Share property to control whether other views from the same mashable or mashup that are added to the workspace can share a *DataTable*. Sharing data, where this is possible, allows views in a workspace to automatically interact when users select a row, bar, slice, location or other object in another view and also ensures that updates to properties affect all views.

  See "Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace" on page 1125 for more information.

You can set any of these properties by editing them for the app in the workspace. See "Edit Workspaces" on page 1126 for information and links to the properties you can set.

This basic app wrapper for the view can also be included in wired interactions within the workspace. It has the same publish and subscription events as any basic app. See "Events and Topics for Wiring Basic and Custom Apps" on page 1178 for more information.

Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace

When you create a basic app using App Maker that contains multiple views for the mashable or mashup, the app has automatic interactions defined for all views that allow a selection event in one view to update the selection in the other views. These automatic interactions are handled by the DataTable that all views in the app share. MashZone NextGen developers should see "About DataTable Events" on page 1527 for more information. Updates to properties for the app, such as pagination, also affect all views in the app.

When you include one view for a mashable or mashup in a workspace, this gets wrapped in one basic app with its own DataTable. You have to explicitly create any interactions with other apps. Updates to properties affect just that app.

Mashboard*automatically* configures wrapper apps for multiple views in a workspace to share a single DataTable in specific cases. The shared DataTable ensures that all the wrapper apps and views participate in automatic selection interactions and updates to properties.

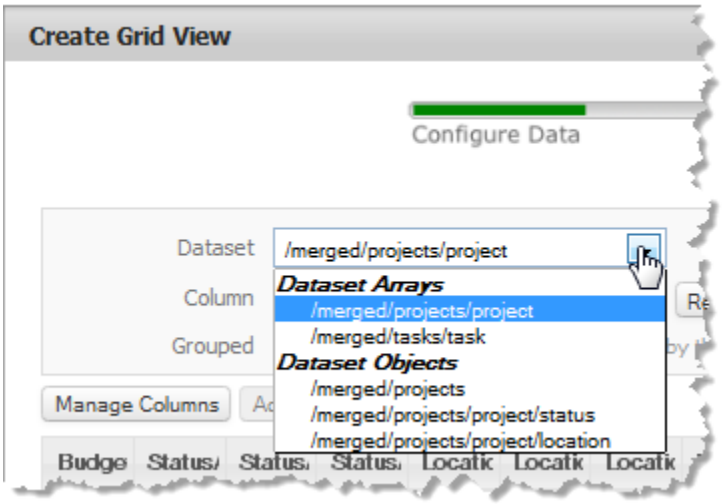
DataTable sharing is enabled for multiple views in a workspace when all of the following are true:

- The views are from the same mashable or mashup.
- The views share the same operation for that mashable or mashup.

  Mashups and many types of mashables have a single operation so this is always true. However, WSDL and database mashables may have several operations.
- The *dataset* selected for these views are identical.

When you add a view, the dataset defines which part of the result provides data to the view. In most cases, this is a path to the repeating items with the data and this is set automatically.

Results for some mashables or mashups may have multiple datasets, however, so that different views for the same operation may have different datasets.



Note: Data sharing is not based on the values of input parameters. Because of this, only one set of views in a workspace can be created with a shared DataTable for a given mashable/mashup, operation and dataset.

You can disable DataTable sharing (and both the selection interactions and property updates) with the **Share Data** property for each wrapper app. See "Set Data Share or Sort and Filter Tools for Views" on page 1131 for more information.

Edit Workspaces

To update or edit a workspace:

1. Find the workspace in the **Workspaces** tab. See "Find Views, Apps or Workspaces in Mashboard" on page 1123 for instructions.
2. Double-click to open the workspace in a design tab.
3. Edit the workspace as needed. For instructions, see:

For Any Workspace	Column or Desktop	Table	Tabbed or Paged
"Copy a Workspace" on page 1128	"Add or Delete Rows and Columns to a Layout" on page 1131		"Add Tabs or Pages" on page 1140

For Any Workspace	Column or Desktop	Table	Tabbed or Paged
"Move, Cut, Copy, Paste or Remove Apps or Views in Workspaces" on page 1127	"Initial Widths and Heights for Column or Table Layouts" on page 1133		"Delete Tabs or Pages" on page 1140
"Control Title Bars and Borders" on page 1128	"Control Padding for Column or Table Layouts" on page 1134		"Change Tab or Page Names" on page 1140
"Set Input Parameters, Pagination, Isolation or Auto-refresh" on page 1130	"Resize Apps in Column or Desktop Layouts" on page 1134	"Change Cell Width or Height with Spanning in Table Layouts" on page 1137	"Change the Order of Pages" on page 1140
"Set Data Share or Sort and Filter Tools for Views" on page 1131	"Set Desktop Layout Colors and Properties" on page 1141	"Fixed Row Heights for Table Layouts" on page 1138	"Set Other Tab or Page Configuration" on page 1140

4. Click **Save**.
   a. Update the description or other properties for this workspace.
   b. Click **Update**.
5. If you have not already done so, you must also Grant Run Permissions to a Workspace to other users to allow them to work with your workspace.

Move, Cut, Copy, Paste or Remove Apps or Views in Workspaces

For workspaces that have more than one column or row, you can move apps to other positions simply by dragging them to the appropriate cell. With the Desktop layout, you can drag apps to any position within the desktop layout.

To remove apps, click **Close** in the app title bar.

You can also move apps (cut and paste or drag and drop) or copy apps *between* two workspaces:

1. Open both the source and destination workspaces, if needed.

2. Click **Configure** in the title bar of the app that you want to move or copy. Choose **Cut** or **Copy** from the Configure menu.
3. Move to the destination workspace and right-click in the cell where you want this app. Choose **Paste** from the menu.
4. Save your changes in the destination workspace, and the source workspace if appropriate.

Copy a Workspace

You cannot edit a workspace that you did not create, unless you are a MashZone NextGen administrator. You can, however, copy any workspace that you have permissions to run and then update your copy:

1. Click **Save As**.
2. Change the name of the workspace, which defaults to `Copy of` *workspace-name* and update any other properties as needed.
3. Click **Create**.

Control Title Bars and Borders

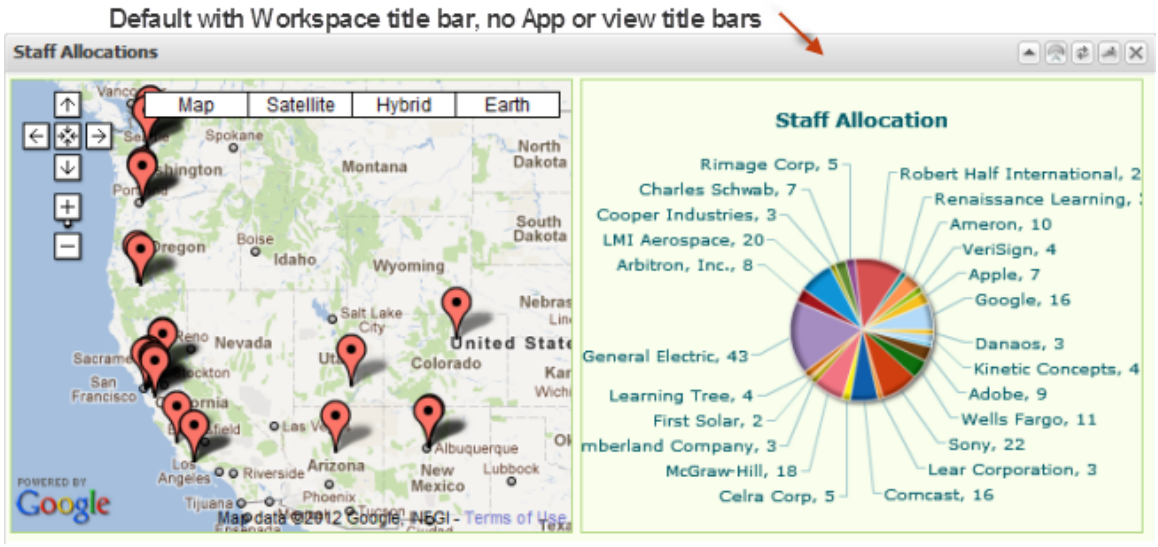
There are several different toolbar buttons or properties that you can configure to control title bars, the buttons within the title bar and the borders around one app in a workspace:

- To show or hide the title bar and/or border for an app in a workspace, chose one of these menu options from the Chrome menu in the Mashboard toolbar:
  - **Chrome** = show both the border and title bar.
  - **No Header** = hide the title bar but show the border.
  - **No Chrome** = hide both the title bar and border.

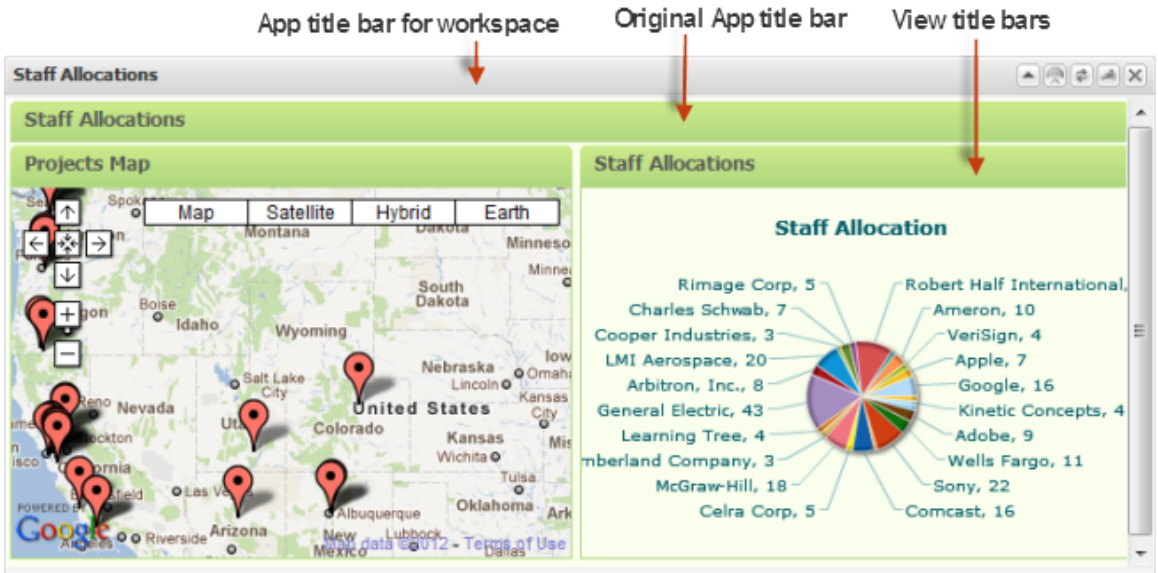
  Important: When you hide the app title bar, users cannot use the toolbar buttons to minimize the app, to update input parameters for the app or to update sort or filter information for the app.

- To show/hide buttons on the toolbar in the app title bar, click **Configure > Properties** in the app title bar and open the **General** tab. Set or clear these properties:
  - **Closable** = clear this option to remove the **Close** button from the title bar for the app.
  - **Draggable** = clear this option to permanently fix the app in this position within the workspace.
  - **Collapsible** = clear this option to remove the **Minimize** and **Maximize** buttons from the title bar for the app.

- To change or hide app titles, click **Configure > Properties** in the app title bar and open the **Properties** tab. Change or set these properties:
  - **App Title** = changes the title that appears in the title bar for the app within this workspace.



  - **Hide App Title** = hides the original title bar of the app. This is set by default because workspaces provide an updated title bar for the app within this workspace. See the following image for an example that shows this original title bar.
  - **Hide View Titles** = for apps with multiple views, this hides the title bars for each view in the app. This option is set by default.



- To change the color of the border for the app, click **Configure > Properties** in the app title bar and open the **Properties** tab. Change the **Theme** property.

  Note: By default, this color only appears in a border around each view within the app. If you show view titles or the original app or view title bar, this color also appears in those title bars.

Set Input Parameters, Pagination, Isolation or Auto-refresh

Click **Configure > Properties** in the app title bar and:

- Use the **Input** tab to update input parameters or to completely hide the input parameter form for an app. This tab is only available if the app has input parameters.

  Note: You can also update input parameters from the **Tools** button in the app title bar.

  You can also set the **Hide Inputs** option to completely hide the input parameter form for the app. This is useful when you provide input parameters through wiring from other apps or forms in the workspace.

- Change pagination in the **Properties** tab with:
  - **Enable Pagination** = whether results should be *paginated* (separated into several pages) for this app.
  - **Rows Per Page** = the number of records that are shown in one page for this app if results are paginated.
- Change the height in the **General** tab with the **Height** property. This sets the height in pixels for this app. The height for the workspace, and row if applicable, do not change when the overall area for the app changes for different devices or orientations. You cannot set the height lower than the minimum height of the app, if it has a minimum height. Changing the height may also add a scrollbar to the app if it does not fit within the designated workspace area.
- If the look or behavior of an app is affected by other apps in the workspace, change app isolation in the **Secure** tab. Simply set the **Secure App** option in the Secure tab.

  Securing an app isolates the app in an <iframe> to ensure that styles or code from other apps in the workspace do not affect this app. Although secure, you can still wire the app to other apps in the workspace so that they work together.
- Change the automatic refresh interval in the **Properties** tab with the **Refresh Interval** property. This is the number of seconds before the app should run any mashables or mashups that is uses and refresh information. Typically this is set to zero indicating that no automatic refresh is required.

  Note: If automatic refreshes are not enabled, you can manually refresh an app using the **Refresh** button in the title bar.

If automatic refreshes are enabled, the **Refresh** button in the title bar toggles automatic refreshes off or on instead.

Set Data Share or Sort and Filter Tools for Views

For the basic apps that wrap views added directly to a workspace, you can control access to sorting and filtering tools and *DataTable* sharing with apps that wrap other views for the same mashable or mashup. Click **Configure > Properties** in the app title bar and:

- Clear or set the **Enable Data Sort** property in the **Properties** tab to hide or show the Sorting tool for other users of this app. This tool is enabled by default.

  See "Sorting and Filtering Properties" on page 1109 for more information on this tool.

- Clear or set the **Enable Data Filter** property in the **Properties** tab to hide or show the Filtering tool for other users of this app. This tool is enabled by default.

  See "Sorting and Filtering Properties" on page 1109 for more information on this tool.

- Clear or set the **Share Data** property in the **Properties** tab to prevent or enable automatic interactions with other apps in this workspace that wrap views for the same mashable or mashup. This also determines whether other app properties, such as theme, are shared.

  See "Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace" on page 1125 for more information on data sharing.

Grant Run Permissions to a Workspace

When you create a workspace, initially only you or a MashZone NextGen administrator may use the workspace. To allow other users to work with the workspaces you create, you or a MashZone NextGen administrator must grant run permissions to other users.

Because workspaces are apps, you grant run permissions to workspaces just as you do to any app, with *one wrinkle*. Users who have permission to use your workspace must also have permission to use each app and each mashable or mashup that is used by these apps. So you may have to update permissions on the apps or other artifact that are used by your workspace.

See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for basic instructions.

Add or Delete Rows and Columns to a Layout

Use the *Column Layout* button or *Table Layout* button in the Mashboard toolbar to change the number of columns (1-16) and/or rows (1-16) in column or table layouts. See "Adding

Columns or Rows" on page 1132 and "Increasing the Maximum Number of Columns or Rows" on page 1133 for instructions.

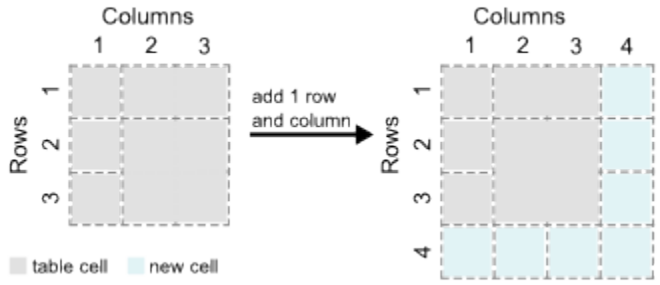
See "Set Size, Spanning and Padding in Column or Table Layouts" on page 1133 for techniques to adjust column or cell sizes and spacing.

With columns, you can also add multiple apps or gadgets to a column. Table cells however only accept only one app or gadget.

Adding Columns or Rows

To add columns or rows, click *Column Layout* or *Table Layout* in the Mashboard toolbar and select more columns or rows for this layout. The maximum available is eight, by default, but you can increase this if needed. See "Increasing the Maximum Number of Columns or Rows" on page 1133 for information.

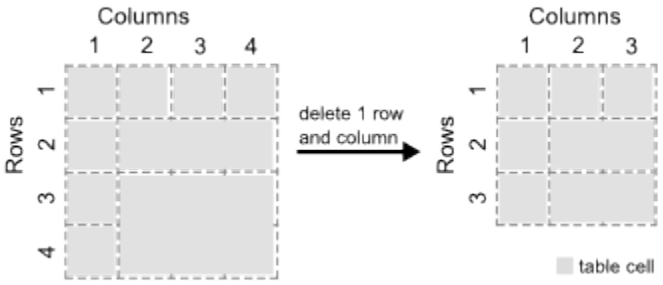
New columns are added on the right side of the workspace. New rows are added on the bottom of the workspace.



For table layouts, this adds one empty cell for each column/row with no spanning.

Deleting Columns or Rows

Before you delete a column or row, *delete* the apps or gadgets in the column(s) or row(s). Mashboard deletes columns from the right and rows from the bottom.



Note: If an app or gadget is spanned across a column or row to be deleted, you do not have to delete the app or gadget before hand. Deleting the column or row simply updates spanning properties to account for the lost cell(s).

Then click *Column Layout* or *Table Layout* and change how many columns or rows are selected.

Increasing the Maximum Number of Columns or Rows

By default, the maximum number of columns or rows you can use in Table or Column layouts is eight. You can increase this maximum up to 16 to get finer grained control of the space and layout of the workspace using Mashboard options. You can also decrease these maximums.

Note: This changes your preferences in Mashboard, but does not affect other users. The preference remains in effect until you change it again.

1. Click **> Options** and select the **Layout** tab.
2. Change the **Max. Rows** and/or **Max. Columns** properties for the Table or Column layout.
3. Click **Apply**.

The next time you click *Column Layout* or *Table Layout*, the maximum number of columns and/or rows that you can select matches these properties.

Set Size, Spanning and Padding in Column or Table Layouts

For column and table layouts, there are several properties and techniques that you can use to properly size and pad the columns or cells for apps and gadgets:

- Initial Widths and Heights for Column or Table Layouts
- Control Padding for Column or Table Layouts
- Resize Apps in Column or Desktop Layouts
- Change Cell Width or Height with Spanning in Table Layouts
- Fixed Row Heights for Table Layouts

You can also use the Hide/Show button for the Mashboard menus to preview changes in the layout full page width. See "Working in Mashboard" on page 1121 for more information.

Initial Widths and Heights for Column or Table Layouts

With column layouts, each column starts out with an equal percentage of the overall workspace width. If you add or delete columns, the width is adjusted automatically. See "Add or Delete Rows and Columns to a Layout" on page 1131 for more information.

You can manually control the percentage of total width for each column. See "Resize Apps in Column or Desktop Layouts" on page 1134 for instructions.

You can set a default initial height for the apps and gadgets that you add to column layouts. Click *Options* in the Mashboard toolbar and set the **Item Height** property. This defaults to 300 pixels.

You can change the height of individual apps, as needed, once they are placed in a column. See "Resize Apps in Column or Desktop Layouts" on page 1134 for instructions.

Table layouts for workspaces in version 3.5 or greater also use an equal percentage of the overall width and height of the workspace to define the width of columns and the height of rows. If you add or delete rows or columns, width and height is adjusted automatically. See "Add or Delete Rows and Columns to a Layout" on page 1131 for more information.

You cannot directly change width and height percentages in table layouts, but you can use spanning to provide more height or width for specific apps or gadgets. See "Change Cell Width or Height with Spanning in Table Layouts" on page 1137. You can also increase the number of rows or columns to get a finer grained grid to support tighter or more flexible layouts. See "Adding Columns or Rows" on page 1132 and "Increasing the Maximum Number of Columns or Rows" on page 1133 for more information.

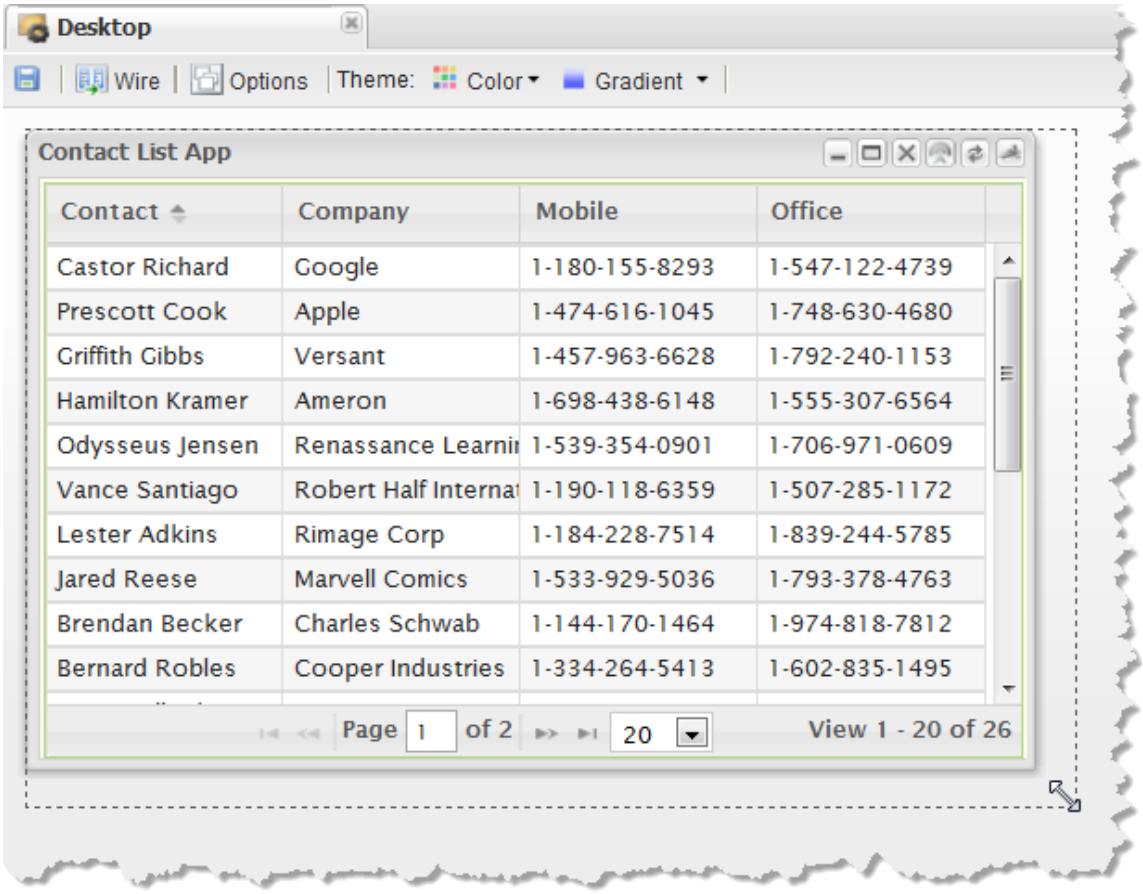
Note: Workspaces with table layouts from versions previous to 3.5 used a fixed height for each row. You can change this to use percentage sizing if desired. You can also choose to use this fixed height for new table layout workspaces. See "Fixed Row Heights for Table Layouts" on page 1138 for more information.

Control Padding for Column or Table Layouts

In addition to the width and height for columns or rows, Mashboard automatically applies a default padding of 5 pixels between columns and rows. You can use the *Spacing* button in the Mashboard toolbar to add or remove padding in 5-pixel increments.

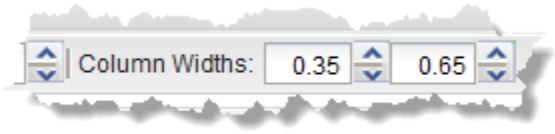
Resize Apps in Column or Desktop Layouts

- With the **Desktop** layout, you can drag any side or corner to update the width and height of each app as well as drag apps to specific positions within the desktop area.

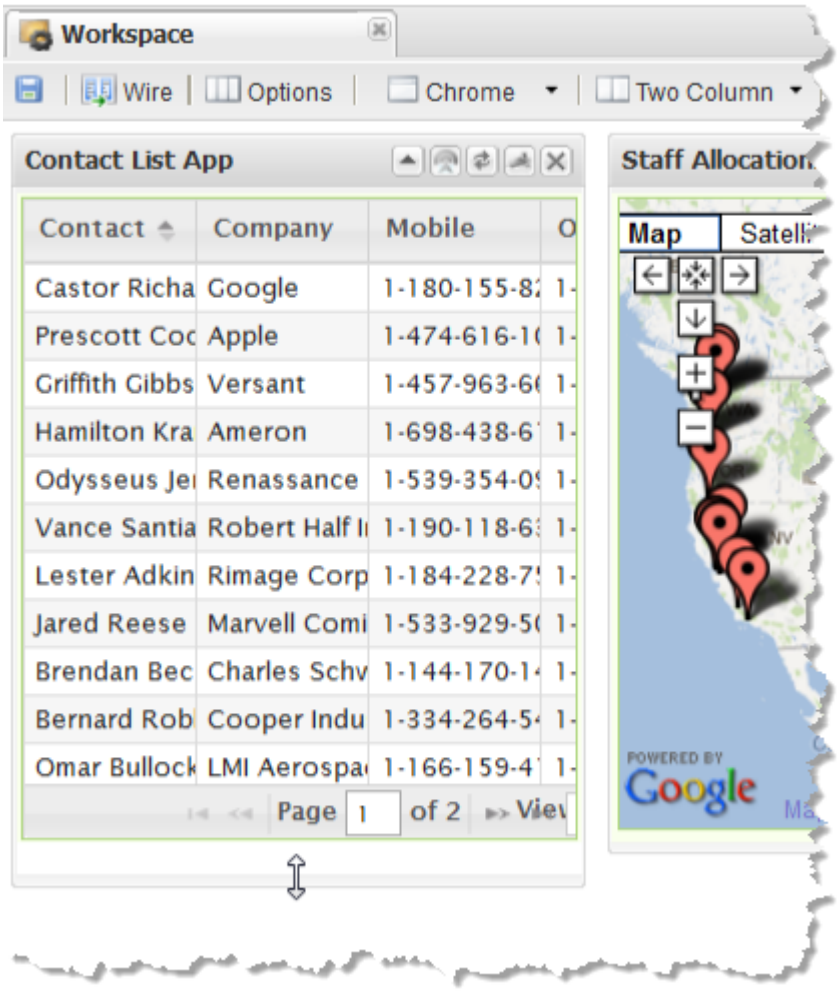


Users may also change the layout when they use the workspace although their changes are not saved between sessions.

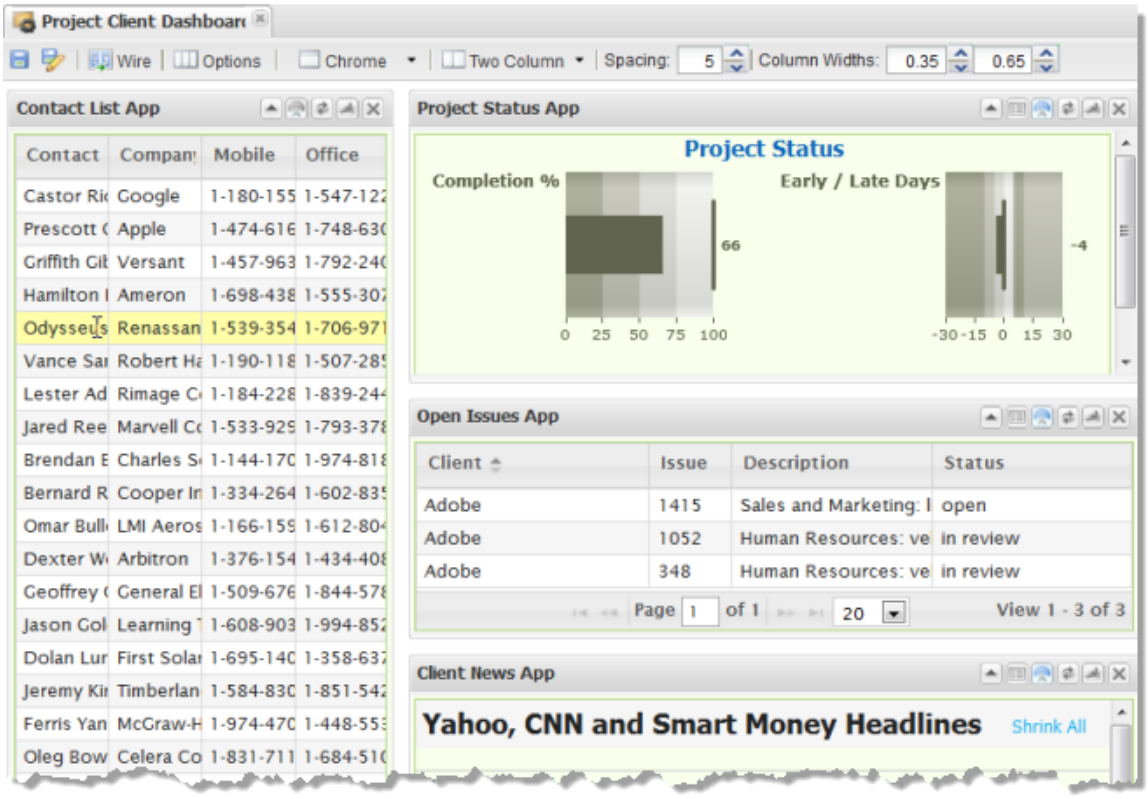
- With the Column layout, you can control the percentage of total workspace width for each column with the **Column Width** options in the Mashboard toolbar:



- You can also change the height of each app by dragging the app's footer, as shown here, or by updating the app's properties.



- You can add multiple apps within a column which can give you a more flexible table-like layout, such as this 2-column layout with three apps in the right column:



Note: Resizing app heights is not available in Table layouts. You can, however, change row spanning properties to adjust the vertical space in Table layouts. See "Change Cell Width or Height with Spanning in Table Layouts" on page 1137 for more information.

Change Cell Width or Height with Spanning in Table Layouts

With table layouts from version 3.5 or later, column and row sizes are a percentage of the overall width and height of the workspace. To make more space available, you use spanning:

1. Add an app or gadget to the top, left cell where you want this content.

   Note: The workspace must have an empty, adjoining cell in the direction you want to span for additional space.

2. If needed, add a column or row to the workspace. See "Add or Delete Rows and Columns to a Layout" on page 1131 for instructions.
3. Click **Configure > Properties** and open the General tab, if needed.
4. To change the width available for this app, increment or decrement the **Column Span** property.

5. To change the height available for this app, increment or decrement the **Row Span** property.

Fixed Row Heights for Table Layouts

Workspaces with table layouts created prior to version 3.5 used a fixed height for table rows. These workspaces can now be mobile-compatible, however, the fixed row height can create problems in mobile devices where size and orientation changes are more challenging. See "Enable Percentage Row Heights for Mobile Compatibility" on page 1138 for instructions.

In other cases, you may want to enable fixed row heights for workspaces with table layouts. See "Enable and Set Fixed Row Heights for Table Layouts" on page 1138 for instructions.

Enable Percentage Row Heights for Mobile Compatibility

To use workspaces with table layouts in mobile devices, you should switch workspaces created in version 3.2.1 or earlier to set row heights as percentages:

1. Click *Options* and open the **Workspace Layout** tab.
2. Clear the **Use Pixel Cell Height** option.

Enable and Set Fixed Row Heights for Table Layouts

If you need to set a fixed row height in a workspace with table layouts:

1. Click *Options* and open the **Workspace Layout** tab.
2. Set the **Use Pixel Cell Height** option.

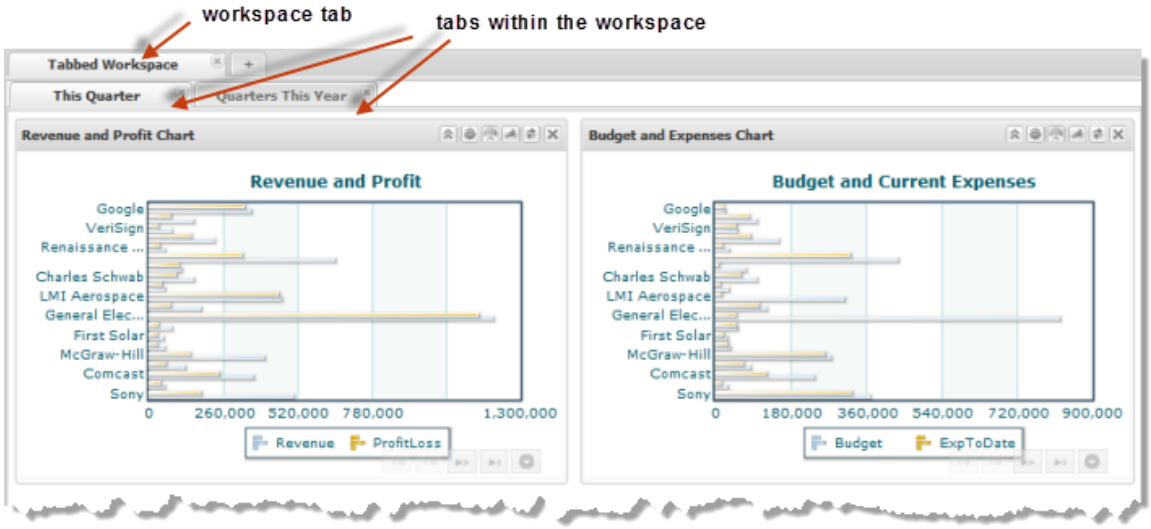
   This option is typically already set for workspaces created in version 3.2.1 or earlier, but is clear for workspaces created in version 3.5 or later.
3. Set the fixed height to use for all rows in the table in the **Cell Height** property. This defaults to 300 pixels.

   This option is only available if the **Use Pixel Cell Height** option is set. Views and apps typically resize to fit in this height, possibly adding scroll bars if needed.

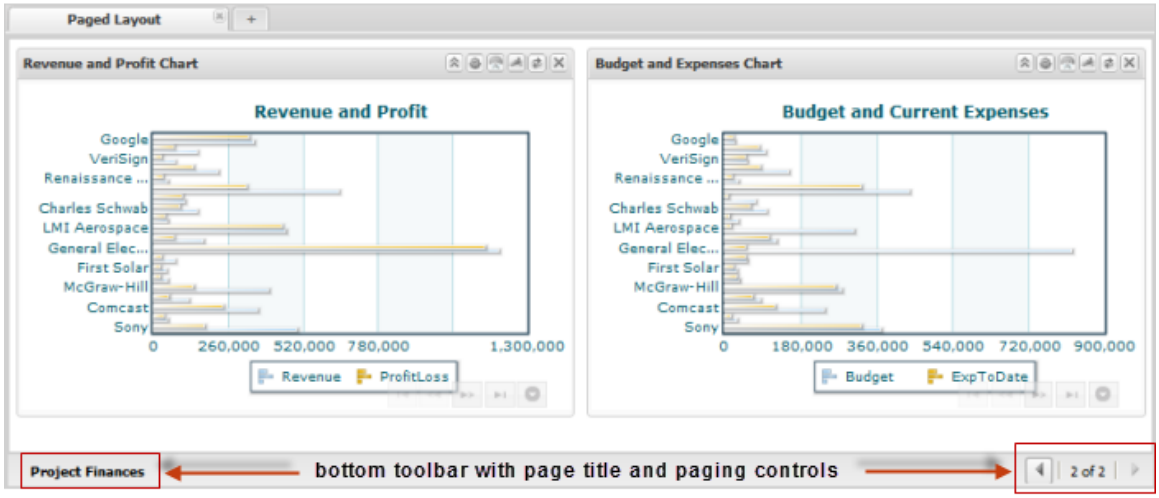
Add, Delete, Arrange or Configure Tabs or Pages in Layouts

Tabbed and Paged layouts are multi-layer workspaces. Each layer (tab or page) has its own layout.

Tabs are stacked, but always visible, providing users direct access to each tab.



While pages are stacked but only one page is visible and accessible at a time, such as this example:



With both of these layouts, you add tabs or pages, choose layouts for each and add apps and gadgets to each. You use options to manage the individual tabs or pages, based on their layout, along with some workspace-wide options:

With Tabbed or Paged workspace layouts, the Mashboard toolbar includes several buttons that allow you to:

- Add Tabs or Pages
- "Delete Tabs or Pages" on page 1140
- Change Tab or Page Names
- Change the Order of Pages
- Set Other Tab or Page Configuration

Add Tabs or Pages

Click *Add Page* or *Add Tab* and then choose the column, table, desktop or web page layout you want to use for that page or tab.

- For column, table or desktop layouts, find and add apps, views or other gadgets as needed. See "Create Workspace Apps with Mashboard" on page 1116 for links to finding content, adding gadgets and other editing tasks.
- To include a web page, see "Web Page Layout for Tabs or Pages" on page 1142 for instructions.

Delete Tabs or Pages

Deleting tabs or pages from a workspace, also deletes the content (apps or gadgets) on that page or tab:

- For tabs, simply click **Close** on the tab to delete that tab.
- For pages, open the page you want to delete and click *Delete Page*.

Change Tab or Page Names

Double-click the tab name or the page name (in the footer) and enter the new name.

Or click *Page Options* or *Tab Options* and change the page or tab title. You can also change the tooltip for the tab or page from options.

Change the Order of Pages

Open the page you want to move and click:

- *Move to Top*
- *Move Forwards*
- *Move Backwards*
- *Move to End*

Set Other Tab or Page Configuration

General options for tabbed or paged layouts allow you to set a variety of options:

- For paged layouts, click *Options*. In the Tab Options tab, set:
  - **Tab Title** for the name of the workspace. See "Change Tab or Page Names" on page 1140 for instructions on changing individual tab names.
  - **Tooltip** for the workspace. See "Change Tab or Page Names" on page 1140 for instructions on changing the tooltip for individual pages.

In the Workspace Layout tab, the **Show Bottom Toolbar** option is set by default. This bottom toolbar contains the name of the current page and the buttons that allow users to move through the pages.

- For tabbed layouts, click *Options*.

  1. In the Tab Options tab, set:
     - **Tab Title** for the current tab open in the workspace. You can also change the name of any tab by double-clicking the name.
     - **Tooltip** for the current tab open in the workspace. You can also set the tooltip for each tab from *Tab Options*.
  2. In the Workspace Layout tab, set:
     - **Tab Width** which defaults to 150 pixels. This defines the width of the curved tab with the tab name, not the width of the content area for the tab. This width applies to all tabs in the workspace.
     - **Allow Tab Rename** which is enabled by default. This allows you to change the tab name.
     - **Allow Tab Close** which is enabled by default. To hide the tab Close button on all tabs, clear this option.

Set Desktop Layout Colors and Properties

The Desktop layout defines a single-layer workspace or defines the layout for one tab or page in a multi-layer workspace. With the Desktop layout, users directly control where apps are positioned as well as their sizes. Users can right-click within the desktop area to tile or cascade app positions. Or they can minimize apps to a *dock* at the bottom of the workspace or open minimized apps.

To define the color theme for desktop layouts using two buttons in the Mashboard toolbar

- **Color**: choose one of the preconfigured colors as the background color for the desktop area of the workspace. For gradient fills, this is the primary color.
- *Gradient*: determines whether the background color is a **gradient** or a solid **plain** fill.

You can also set the initial size to use for apps added to the desktop:

1. Click **Options** and open the **Workspace Layout** tab.
2. Update the **Default Width** and **Default Height** properties as needed. These default to 500 by 400 pixels respectively.

Web Page Layout for Tabs or Pages

You can use this workspace layout *only* as one tab or one page in a multi-layer workspace. This layout allows you to pull in any accessible web page that you specify by URL.

See also "Add Web Pages to a Workspace" on page 1173 for other ways to add web pages to workspaces.

For tabs or pages that use this layout, you add a tab or page and choose this layout. Then enter the URL to the web page to appear in this tab or page.

You can set the following layout properties from the Page Options or Tab Options button:

- **Page or Tab Title**
- **Page or Tab Tooltip**
- **URL** for the web page.
- **Enable Scrolling** which is disabled by default. Enabling this adds scroll bars within the page.

Add a Real-Time Line Chart

The Real-Time Line Chart plots lines for event mashables and automatically handles ongoing updates for events as they occur. For more information, see "What are real-time charts?" on page 74.

To add a real-time line chart

1. Open the **Palette** tab, and if needed open an existing workspace or start a new one.
2. Drag and drop the **Line Chart** utility into the location you want in an open workspace.
3. To configure your chart, click **Configuration** on the toolbar and complete the wizard. See "Configure the Real-Time Line Chart" on page 1142 for details.

Configure the Real-Time Line Chart

The configuration wizard opens with the first step to Select a Data Source. Then, you can:

- Define Date Columns
- Assign Columns to Axes
- Set Column Properties
- Set Chart Properties

Select a Data Source

To find and select the event mashable or Apama mashable that you want to use as the data source for this chart, either:

- Select the event source for the mashable you want from the list and click **Next**.

  Or

- Enter part of the name for the event source to filter the list. Select the event source you want and click **Next**.

Select a Data Source

To find and select the event mashable or Apama mashable that you want to use as the data source for this chart, either:

- Select the event source for the mashable you want from the list and click **Next**.

  Or

- Enter part of the name for the event source to filter the list. Select the event source you want and click **Next**.

Define Date Columns

If none of the columns for this event or Apama mashable are dates or times, click **Next**.

Note: Date configuration information is not required for the `Event_LastUpdateTime_Internal` column included in Apama mashable events.

If there are columns that are dates:

1. Set the **Date** option for that column.

   If the data for this column is numeric, MashZone NextGen treats this as the number of milliseconds since January 1, 1970 to determine a date and time.

2. If the data for a date column is text, you must also:

   a. Enter a pattern as the **Format** for this data to allow MashZone NextGen to correctly interpret the date. Use the following characters to define the pattern:

Character	Meaning
y or Y	Year If the data contains a four digit year, the format pattern can contain any number of `y` characters to recognize the year component of the date.

Character	Meaning
	If the data contains only a one or two digit year, a format pattern of `yy` returns a four digit year while a format pattern of `yyyy` returns just the year.
M	Month If the date has a text abbreviation of the month, use `MMM` as the format pattern. If the month is numeric, use `MM` or `M` as the format pattern.
d or D	Day
h or H	Hour
m	Minute
s or S	Second or Milliseconds
a or A	AM/PM
- or /	As the date component separator
:	As the time component separator
space or other literals	Can be used within the format pattern and appear exactly as typed.

b. Select the **Language** for this data.

3. Once you have identified all columns that are dates or times, click **Next**.

Assign Columns to Axes

You assign event columns to the axes of the chart and optionally partition events into subcategories to determine how event data will be plotted in the chart. To assign a column, select one of the columns from the list on the left and drop it into a field. To delete a column drag the column out of a field and drop it.

With measure columns that you assign to the Y axis, you must also choose an aggregate function to apply to the measure: `Sum`, `Min`, `Max`, `Avg` or `Count`.

There are three patterns you can follow to assign columns:

- **Dynamic Time Line for One or More Measure Across All Events** aggregates and plots all events along a growing time line, as events occur. Each measure you choose to include generates one line.

  To create a dynamic time line across all events:

  - Assign a date column to the **X-axis**.

    For Apama mashables, this is the `Event_LastUpdateTime_Internal` column.

  - Assign one or more numeric columns to the **Y-axis**. Choose an aggregate function to apply to each column to determine how event data is combined.
  - Skip the **Partition** field.

- **Dynamic Time Line for One Measure Partitioned by Category** plots all events on a growing time line. Events are partitioned by a category, each unique category is plotted as a separate line for one measure.

  To create a dynamic, partitioned time line chart:

  - Assign a date column to the **X-axis**.

    For Apama mashables, this is the `Event_LastUpdateTime_Internal` column.

  - Assign a single numeric column to the **Y-axis** and choose the aggregate function to apply.
  - Assign a text column to the **Partition** field. This column defines the partitions.

- **Dynamic Categories** plots event data across categories on the X axis. There is no timeline, but as events occur, the lines on the chart are completely replotted.

  For example, you can plot temperature, humidity and other weather measures for different cities using the Apama Weather sample. The category is city.

  To create a dynamic categories chart:

  - Assign a text column to the **X-axis**. The unique values of this column represent one point on the line.
  - Assign one or more numeric columns as measures to the **Y-axis**. Each column generates a separate line.

    Choose an aggregate function to apply to each measure.

  - Skip **Partition**.

Set Column Properties

You can change labels and visual formats for any column:

- Set a **New Name** to change the label that appears in tooltips and the chart legend, if any.
- Select a **Format** pattern for numeric or enter a pattern for date columns. You can:

- Choose a pattern to set the decimal and thousand place separators as well as the total number of decimal places for numeric columns.

  For example, choose:

  `1,234.56`

  To see numbers with two decimal places, a period as the decimal point and a comma as the thousand separator.

- Set the format and separators to use with dates and times.

  Use these symbols:

Character	Meaning
y or Y	Year If the data contains a four digit year, the format pattern can contain any number of `y` characters to recognize the year component of the date. If the data contains only a one or two digit year, a format pattern of `yy` returns a four digit year while a format pattern of `yyyy` returns just the year.
M	Month If the date has a text abbreviation of the month, use `MMM` as the format pattern. If the month is numeric, use `MM` or `M` as the format pattern.
d or D	Day
h or H	Hour
m	Minute
s or S	Second or Milliseconds
a or A	AM/PM
- or /	As the date component separator
:	As the time component separator
space or	Can be used within the format pattern and appear exactly as typed.

Character	Meaning
other literals	

- Turn on rounding for numeric columns to round values to zero decimals.

Click **Next**.

Set Chart Properties

Finally, set general properties for the chart including:

- Enter a **Chart title** if desired.
- Change the default value of the **X-axis name** to display a label for this axis. This defaults to the label for the column assigned to this axis.

  You can also delete the value to remove this label from the chart.
- Change the default value of the **Y-axis name** to display a label for this axis. If you assigned a single measure to this axis, this defaults to the label for that measure. If there are multiple measures for this axis, the label defaults to `Value`.

  You can also delete the value to remove this label from the chart.
- Choose the **Line type**:
  - `Linear`
  - `Curved`
  - `Step-after`
- Set **Legend** to display a legend for the chart.

Click **Save**. The chart begins to plot data and updates periodically as events occur.

Add a PPM Chart to a Workspace

Process Performance Manager (PPM) lets you discover and analyze processes in your organization, whether they are managed by a business process management solution (*BPMS*) or not. Using data sources throughout your enterprise, PPM can model a process and assess its performance across a variety of dimensions. PPM can also mine other data in your enterprise for meaningful patterns, trends, or correlations.

You can include insights from the analyses you perform in PPM in a workspace using the PPM Chart app in MashZone NextGen. This app allows you to add charts that you have created and saved as favorites in PPM to a MashZone NextGen workspace.

Note: The PPM Chart app is only available in MashZone NextGen if:

- A MashZone NextGen administrator has imported this app into MashZone NextGen, configured a connection to the Integrated MashZone

Server for MashZone NextGen and configured a connection to the PPM client that hosts the charts you want to work with.

- PPM is installed, configured and PPM clients are accessible to MashZone NextGen.

1. In the PPM client that hosts the analysis you want to work with, find the favorite for the chart you want to use and:
   a. Right-click the favorite.
   b. Select **Use in dashboard** to copy the URL for this favorite to the clip board.

   Note: You can use most chart types from PPM in MashZone NextGen*except* for the following:
   - Assessment charts
   - Communications network charts
   - EPC (event-driven process chain) models
   - Relational explorer charts

2. In Mashboard in MashZone NextGen, create or open the workspace where you want to use this PPM chart.
3. Find the PPM Chart app in the Apps menu and drag it into the desired position in the open workspace.

   See "Find Views, Apps or Workspaces in Mashboard" on page 1123 for more information.

   The PPM Chart app loads and displays a button that you must use to identify the specific PPM chart you want to display in this workspace.
4. Click **Select PPM Favorite** in the app.
5. Paste the PPM favorite URL that you copied earlier in the **URL** field.
6. Click **Next**.

   Information about connecting to this chart using the URL for this favorite displays, including the name of the connection to PPM (the **Data source**).

   Note: If no data source is shown, cancel this configuration for the chart. Ask a MashZone NextGen administrator to add a connection to the PPM client where you found this favorite and then return to configure this PPM chart.
7. Enter the user name and password to use for PPM in the **User** and **Password** fields.
8. Click **Apply**.

   The configuration window closes and the PPM chart you have selected displays in the workspace.

Additional PPM Chart Options

In addition to standard app properties, the PPM Chart app has additional configuration options that you can use to update the favorite, allow users to go directly to PPM from the chart and control other visual attributes. See "Update a PPM Chart" on page 1149 for more information on these custom properties.

You can also wire the PPM Chart app to interact with other apps in the workspace for these interactions:

- *Click*: to publish user selections within this chart. Use standard wiring techniques to enable this interaction. See "Wiring App Interactions in a Workspace" on page 1175 for instructions.
- *Select*: to set a selection within the chart based on user actions in other apps within the workspace. Use standard wiring techniques to enable this interaction.
- *Filter*: to apply filtering to the data the chart displays based on user actions in other apps within the workspace.

  To support wired filters, you must first define a filter for the PPM Chart app. See "Wire Selection and Filter Events for PPM Charts" on page 1150 for instructions.

  Then use standard wiring techniques to enable this interaction.

Update a PPM Chart

In addition to the standard properties for apps in a workspace, the PPM Chart app has custom properties and actions that you or other users can use to customize or update the chart when it is in a workspace.

See "Edit Workspaces" on page 1126 for more information on standard properties for apps in a workspace.

See the following sections for information on using the custom properties for the PPM Chart app:

- Update the PPM Chart Favorite
- Update PPM Chart Selection Properties
- Control User Jumps to PPM

See also "Wire Selection and Filter Events for PPM Charts" on page 1150 for information on providing data filtering in PPM charts that is wired to user actions in other apps in a workspace.

Update the PPM Chart Favorite

You can change the chart that is shown in a PPM Chart app in a workspace using custom actions available from the Configure menu:

1. Click **Configure** in the title bar of the PPM Chart app that you want to update favorite information for.
2. Select **Select favorite**.
3. Update the Data source, Favorite URL, Language or credentials used for authentication.
4. Click **Apply**.

Update PPM Chart Selection Properties

Custom properties let you control the color and how opaque the highlight is that is used to show a selection when users click in a PPM Chart:

1. Click **Configure** in the title bar of the PPM Chart app.
2. Select **Properties** and click the **Properties** tab.
3. If desired, change how opaque the select highlight color is in the **Select Opacity** property. Higher values make the color more opaque.
4. If desired, enter a hexadecimal code for the highlight color to use to indicate selection. The default color, `0000ee`, is a very pale blue.
5. Click **Apply**.

Control User Jumps to PPM

By default, users who work with PPM charts within the PPM Chart app can highlight information and see details within the chart, but do not have direct access to jump to PPM and work directly with this favorite.

You can choose to show a link in the PPM favorite to provide direct access to PPM using a custom configuration property for the PPM Chart app:

1. Click **Configure** in the title bar of the PPM Chart app.
2. Select **Properties** and click the **Properties** tab.
3. Set the **Allow Jump to PPM** option to display a link within this chart. Or clear this option to remove the link.
4. Click **Apply**.

Wire Selection and Filter Events for PPM Charts

The PPM Chart app is a custom app available in MashZone NextGen to integrate charts that users create in PPM into workspace apps. This custom app supports loosley-coupled interactions with other apps in a workspace using the following events:

- *Click*: to publish user selections within this chart. Use standard wiring techniques to enable this interaction. See "Wiring App Interactions in a Workspace" on page 1175 for instructions.

- *Select*: to set a selection within the chart based on user actions in other apps within the workspace. Use standard wiring techniques to enable this interaction.
- *Filter*: to filter data in the chart that is displayed in the PPM Chart app based on user actions in other apps within the workspace.

  To support this custom event and subscription, you must define a filter to use with this chart. This filter configuration also defines the subscription payload for this event as a set of properties.

  See "Set Filter Properties for Wiring PPM Charts" on page 1151 for instructions and "Filter Wiring Example for a PPM Chart App" on page 1155 for an example.

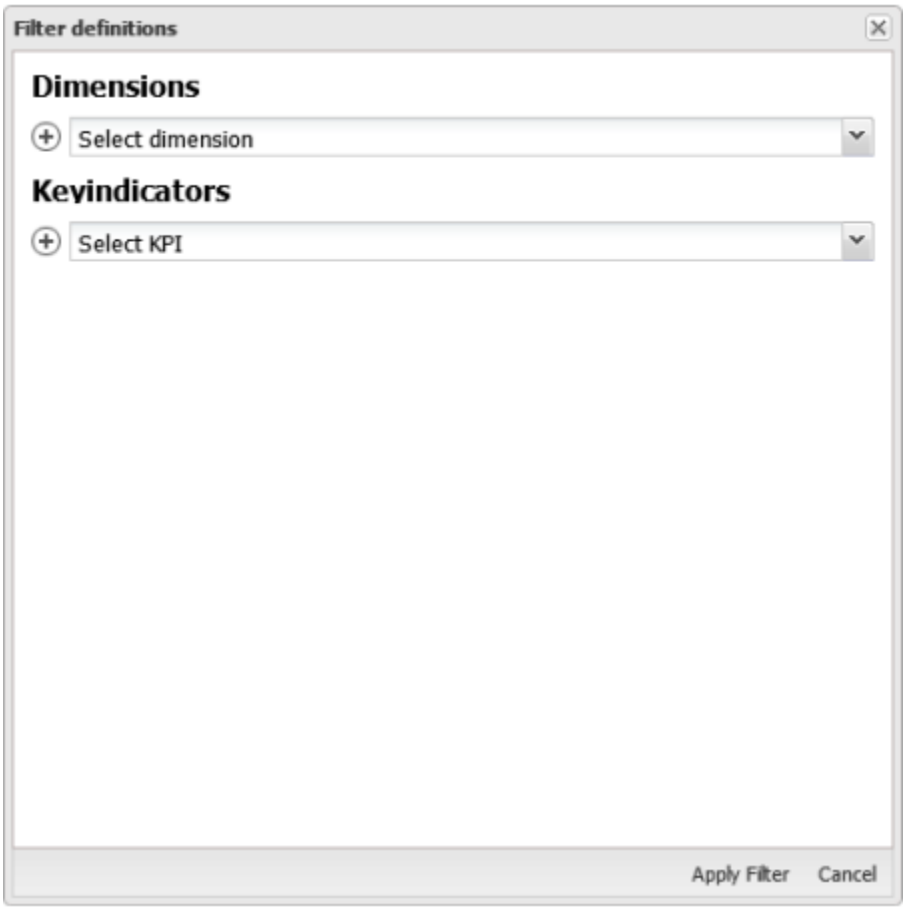
Set Filter Properties for Wiring PPM Charts

Filters defined in a PPM Chart app are wired to interactions in another apps within the workspace rather than simply providing a form that users fill in. See "About PPM Filters" on page 1158 for more information on filter criteria and options.

To define a filter for wiring

1. Click **Configure** in the title bar of the PPM Chart app.
2. Select **Define filter**.

   The Filter definitions window opens with an empty Dimension criteria and an empty Key Indicator criteria.



3. Add at least one filter criteria for a dimension or key indicator. For instructions, see:
   - "Add a Dimension or Key Indicator Criteria" on page 1153

     You can add any number of criteria. Each criteria defines one filter property that can be wired. With multi-level dimensions, you must also define which levels should be tested.
   - "Add Additional Criteria for an Existing Dimension or Key Indicator" on page 1154

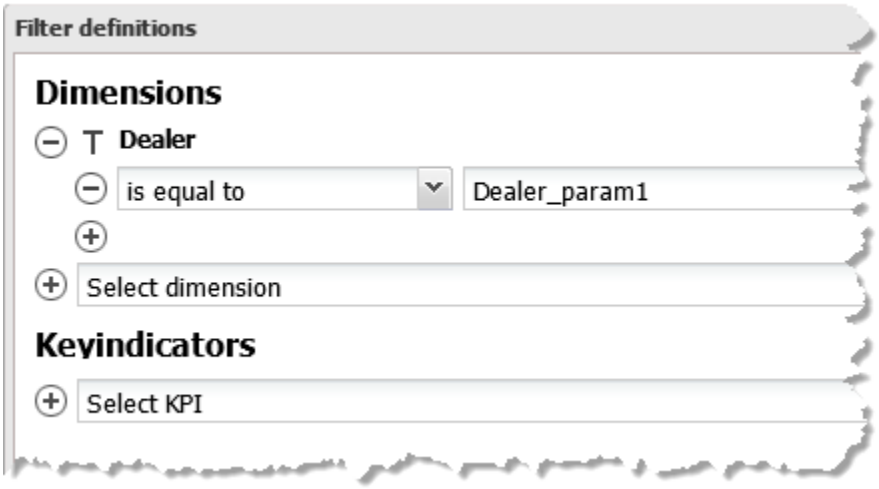
     With multiple criteria for one dimension, you must also determine whether all criteria must be met or any one criteria is sufficient.
4. Once the filter is fully defined, click **Apply Filter**
5. With the filter defined for a PPM Chart app, complete wiring to enable the filter interaction and save the workspace.

   See "Wiring App Interactions in a Workspace" on page 1175 for basic instructions on wiring interactions between apps in a workspace. See "Filter Wiring Example for a PPM Chart App" on page 1155 for an example of wiring filter interactions for a PPM Chart app.

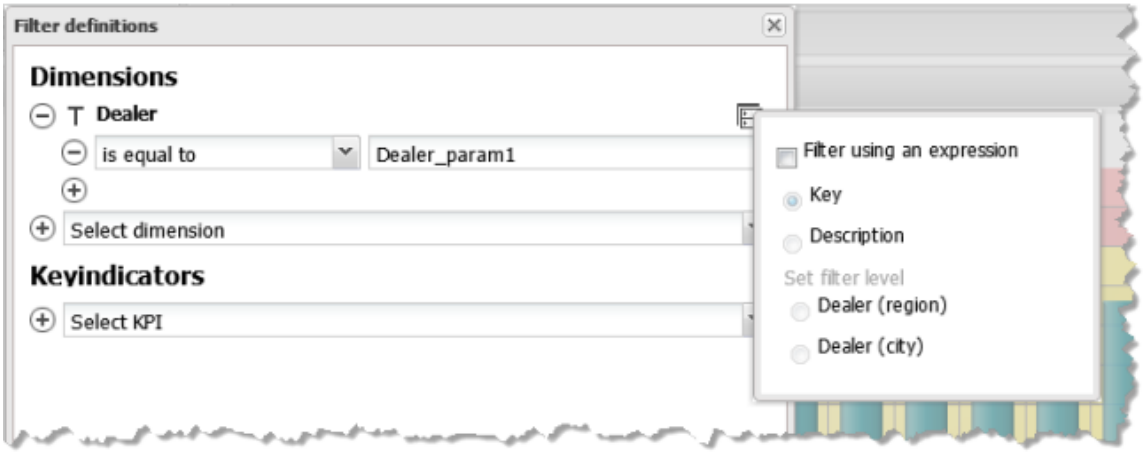
Add a Dimension or Key Indicator Criteria

1. Select either a dimension or a key indicator to use for this filter criterion.

   A comparison operator and property field display to let you finish this criterion.



2. Change the comparison operator, which defaults to `is equal to` or `is less than`, if needed.
3. If desired, change the default name of the filter property for this criterion.



   This third field identifies the property for this filter that will be wired to published events from another app. The property name defaults to *dimension-or-key-name* `_param`*n* .

   Property names cannot contain spaces and generally should not contain punctuation or symbols other than dash (-) or underscore (_).

4. If this criteria should compare the description for this dimension or key indicator rather than the key value:

a. Click .

b. Set the **Filter using an expression** option.

c. Set the **Description** option.

5. If this is a multi-level dimension, set the dimension level:

a. Click .

b. Set the **Filter using an expression** option.

c. If needed, set the **Description** option to have this criteria compare the description rather than the key value.

d. Choose the appropriate **Set filter level** option.

6. If needed, add more criteria for this dimension or key indicator or for other dimensions or key indicators.

Add Additional Criteria for an Existing Dimension or Key Indicator

You can define two or more criteria for a single dimension or key indicator and determine whether all criteria must be met (AND) or any one criteria (OR) is sufficient.

1. In the existing Dimension or Key Indicator rule, click ⊕.

   Another comparison operator and property name field display.

2. Change the comparison operator, which defaults to `is equal to` or `is less than`, if needed.

3. If desired, change the default name of the filter property for this criterion.

   This third field identifies the property for this filter that will be wired to published events from another app. The property name defaults to *dimension-or-key-name* `_param`*n* .

   Property names cannot contain spaces and generally should not contain punctuation or symbols other than dash (-) or underscore (_).

4. If this criteria should compare the description for this dimension or key indicator rather than the key value:

a. Click .

b. Set the **Filter using an expression** option.

c. If needed, set the **Description** option to have this criteria compare the description rather than the key value.

5. If this is a multi-level dimension, set the dimension level:

a. Click .

b. Set the **Filter using an expression** option.

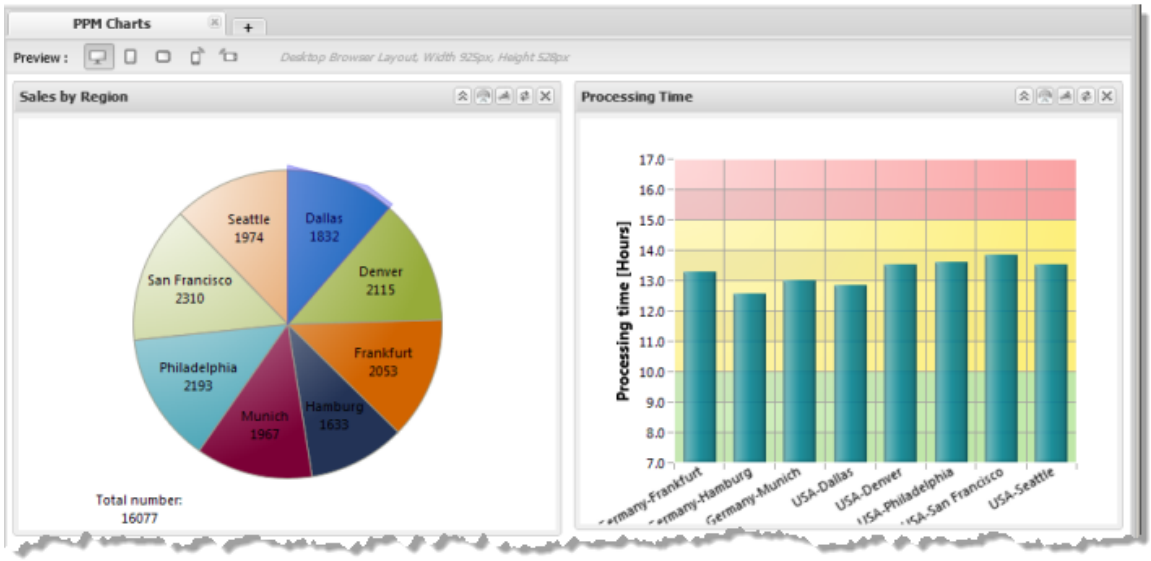
c. If needed, set the **Description** option to have this criteria compare the description rather than the key value.

d. Choose the appropriate **Set filter level** option.

6. If needed, add more criteria to this dimension or key indicator.
7. By default, the **AND-link conditions** option is set indicating that all of these criteria must be met for this dimension or key indicator. If any one criteria is sufficient, clear this option.

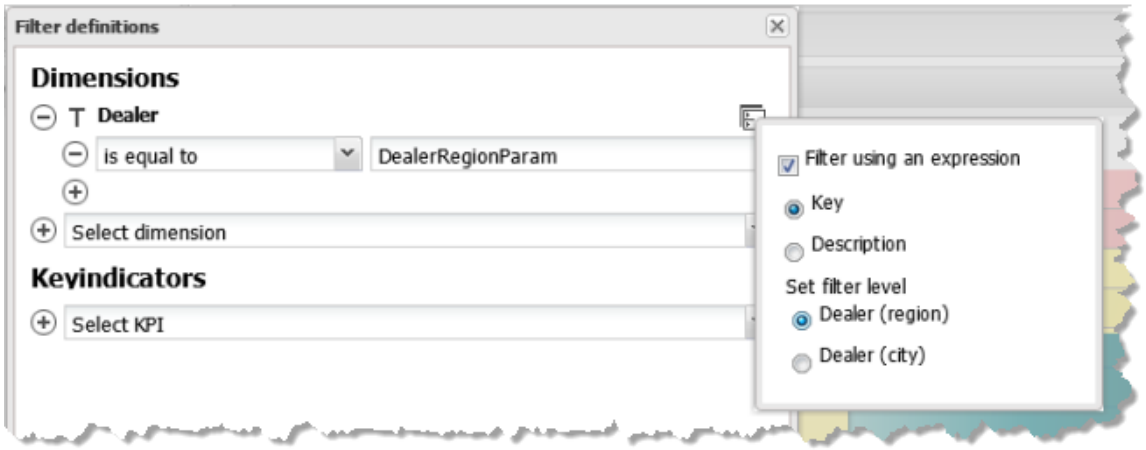
Filter Wiring Example for a PPM Chart App

This example is a workspace using a two-column layout and the PPM Chart App in both columns pulling in two charts:

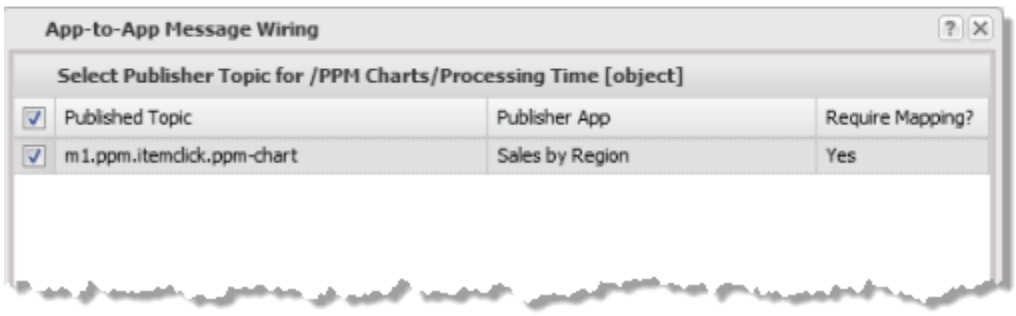


The Sales by Region chart summarizes sales across a two-level Dealer dimension with Region as the top level and City as the subordinate level. The Processing Time chart shows order processing time for the same two-level Dealer dimension. The default, unfiltered view of Processing Time shows the average time for all region/city combinations.

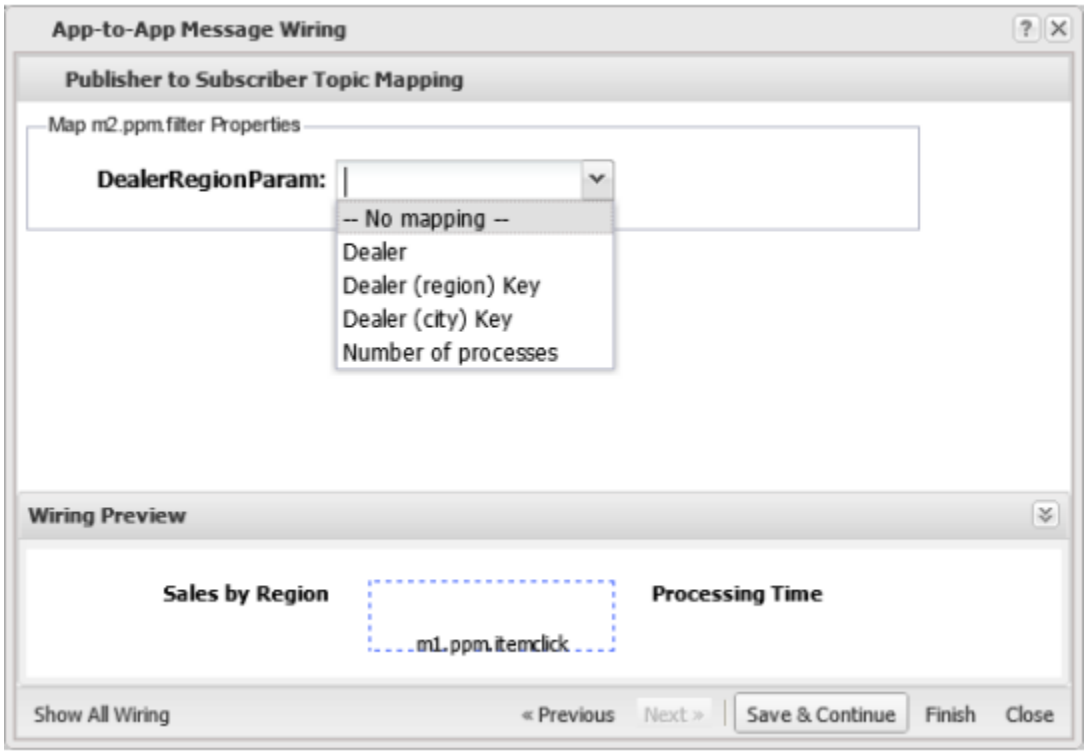
The filter is defined on Processing Time to filter the Dealer dimension with an expression to compare only the Region level:



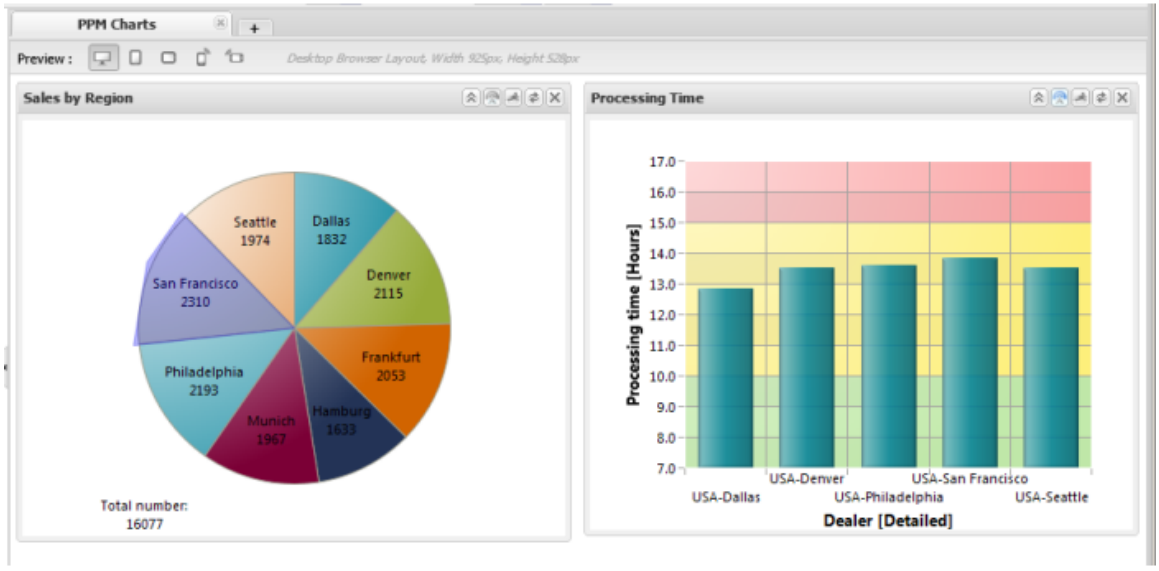
The wiring that enables the filter interaction, uses the Filter subscription topic for Processing Time and the Item Click publish topic for Sales Region.



Mapping for the subscription topic payload, maps the Dealer (region) from the published topic to the DealerRegionParam parameter for Processing Time defined in the filter.



Once wiring is complete, clicking on any pie slice in Region Sales filters the data shown for Processing Time to those cities that belong to the matching region. In this example, San Francisco is selected which filters Processing Time to show the USA region:



About PPM Filters

A PPM filter consists of one or more criteria each of which compares the values for a *dimension* or a *key indicator* in the chart to data from an event published by another app.

Key indicators are also sometimes called *measures* because they are always numeric data - they are the things you measure in the chart, such as quantity or cost.

Dimensions contain *categorical data*, the categories or other attributes of the data that are differentiators for the analysis such as region, city or customer. Dimensions can also contain date and time information, such as the order date.

Dimensions in PPM can also have one or more levels. Single level dimensions have simple data, such as US, Argentina or Germany for the region. Multi-level dimensions have complex data, such as US-Denver for the region and city.

Dimensions also have a key, the value for that record such as Denver for the city, and potentially also a description. Criteria for a filter can make comparisons against the key or against the description.

The comparison to determine what data to show in the filtered chart depends on the *comparison operator* you define in the criteria. The operators that you can use in a filter criteria depend on what type of data the dimension or key indicator contains.

See "Set Filter Properties for Wiring PPM Charts" on page 1151 for an example.

Add a Drill Down App in a Workspace

A drill down app is similar to the paged layout, but it layers two or more apps, stacked on top of each other. As users page or click through, the next layer down overlays the current layer.

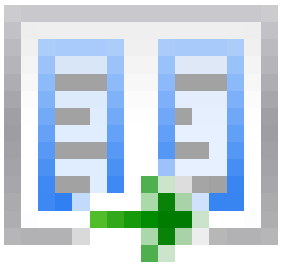
One common use case for drill down is *master-detail*. Users select one item on the current layer to open the next layer down with details about that selected item. Drill down, however, is not limited to master-detail relationships. You can use it to easily group any related apps and views.

With drill down apps, users can 'flip' pages using the pagination toolbar. It is also very common to add wiring to a drill down apps to allow users to click within each page to open the next page down.

To create a drill down app in a workspace

1. Click the **Palette** menu and expand **Utilities**.
2. Drag **Drill Down** into the workspace and drop it into the position you want it.

   This creates two pages initially.
3. If desired, double-click the **Drill Down** label to change the title.
4. Drag and drop an app or view into the first and second pages.

5. Add more pages if needed:
   a. Click **Configure >** **Add Page** in the title bar to add a page at the end of the drill down.
   b. Drag and drop an app or view into the new page.
6. To enable click-through for the drill down (users click in each page to open the next one), add a wire between the apps on adjacent pages:
   a. Click  **Wire** to open the App Wiring wizard.
   b. Create a wire for each adjacent pair of pages from beginning to end, subscribing for an event on the lower page to a publish event on the higher page. See "Wiring App Interactions in a Workspace" on page 1175 for instructions on adding wires.

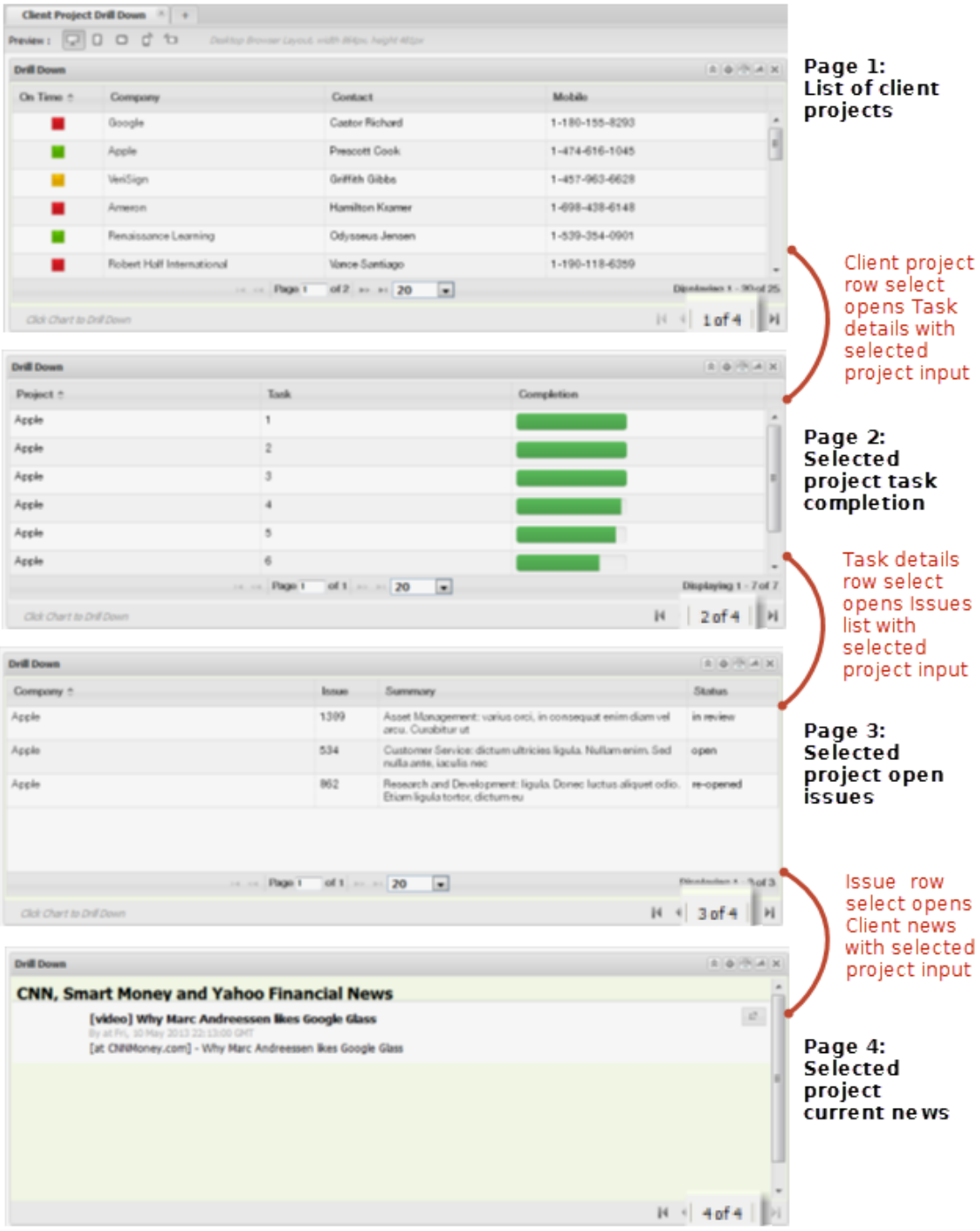
      In many cases the `propertychange` event is used for the subscriber to pass some information used to select the detailed information for the lower page. Publish events can be any appropriate click event.

      See the "Example 137. Drill Down Example" on page 1159 section for an example.
   c. After you have created and saved all the wiring, close the App Wiring wizard.
7. Test the wires for each step of the drill down.
8. Complete and then save the workspace.

Drill Down Example

This example is a four page drill down:

1. *Contact List:* a grid with a list of open projects. Click on any row to drill down.
2. *Task Completion Details:* a grid with tasks for the project selected from Contact List. Click any task to drill down.
3. *Open Issues:* a grid with issues for the project selected from Contact List and passed by Task Detail. Click any issue to drill down.
4. *Client News:* with current news for the client selected in Contact List and passed by Open Issues.



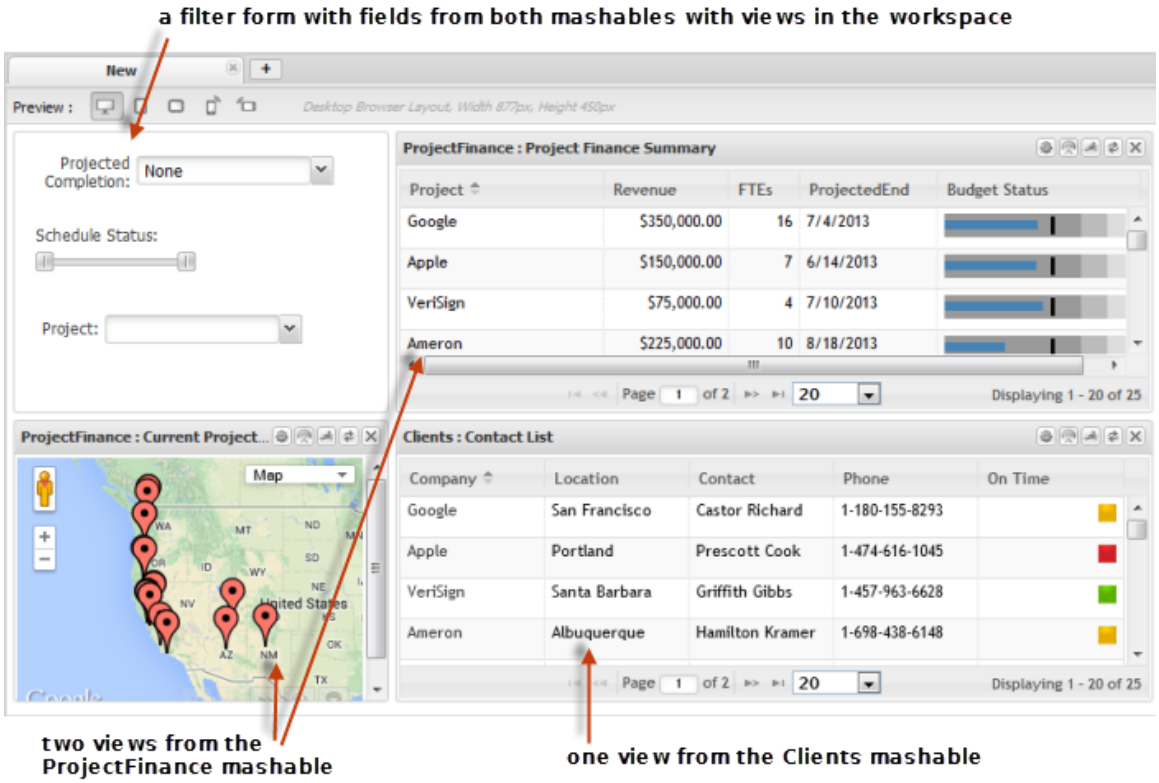
This uses three wiring rules to create the drill down behavior:

- Task Details.propertychange subscribes to Contact List.datatable.rowselect
- Open Issues.propertychange subscribes to Task Detail.datatable.rowselect
- Client News.propertychange subscribes to Open Issues.datatable.rowselect

Add a Filter for Apps in a Workspace

You can add a form to your workspace that allows users to filter the information shown in one or more apps within the workspace. Filters can be added regardless of whether the underlying mashable or mashup has input parameters that also perform filtering.

The example shown here has three views based on two mashables and a filter form:

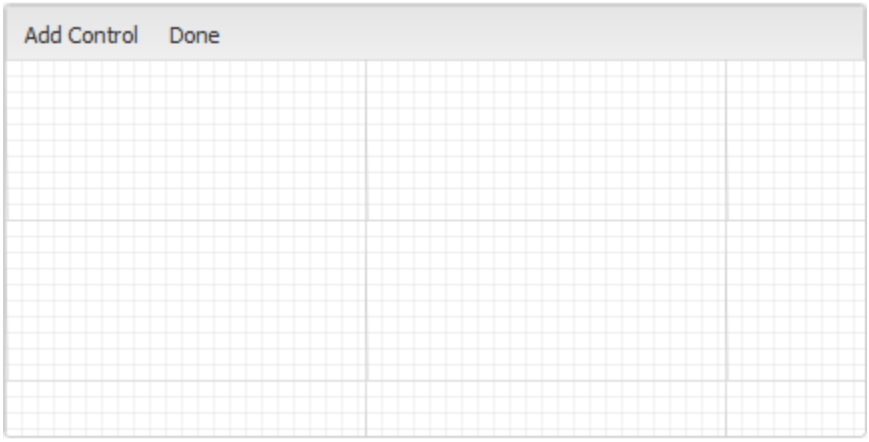


You define the individual fields or other controls to include in the filter form based on the fields available in any app in the workspace. You also control the placement of controls to make the filter form easy to use.

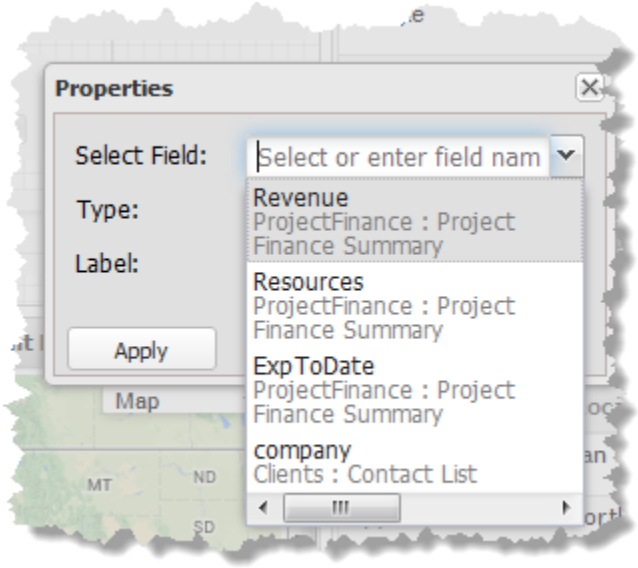
To add a filter form to a workspace

1. Start a new workspace or open an existing workspace in Mashboard.
2. Add all the apps and views that should be in this workspace.
3. Select the **Palette** menu, expand the Controls section and drag **Filter Controls** into the location where you want to filter to be placed in the workspace.

   An editing canvas opens where you can add controls and manage the layout of the filter form.



4. Click **Add Control** to add one field or other type of control for filtering one or more views or apps within the workspace.
5. Select the field to be filtered by this control. The list includes every column in data from every mashable or mashup used in this workspace.



6. Choose the type of control to include in the form for this field. The types that are available depend on the type of data for the field you have selected:
   - `Drop down`: a list of available values drops down. Users can only select from these values. Use this only with columns that have a limited number of possible values and very long drop down lists are hard to use.

     This type of control is available for fields of any datatype.
   - `Radio Button`: one radio button is provided for each possible value for this field. Use this only with columns that have a very limited number of possible values to ensure that users can easily find a specific value.

This type of control is available for fields of any datatype.

- `Slider`: a control with upper and lower limit sliding buttons to allow users to select a range of numeric values. This provides a continuous range to play with but can be difficult to have fine grained control of the range.

  This type of control is available only for fields with numeric data.

- `Text`: users must type in the value to use for filtering. This is the most flexible control, allowing filtering to work with fields that have very large possible values.

  This type of control is available for fields of any datatype.

7. Update the default **Label**, if needed, and click **Apply**.
8. Drag the control within the canvas to move it to the preferred location. Or expand the default size of the control.
9. Add more controls, as needed. Click **Done** once the form is complete and satisfactory.

The completed form displays. Test the controls to see the automatic interactions in the various apps within the workspace.

See also "Edit or Delete Filter Controls" on page 1163.

Edit or Delete Filter Controls

You can edit the properties of individual controls of a filter form, change the placement of controls or delete controls.

1. Right-click anywhere in the filter form and select **Properties**.

   The form editing canvas opens.

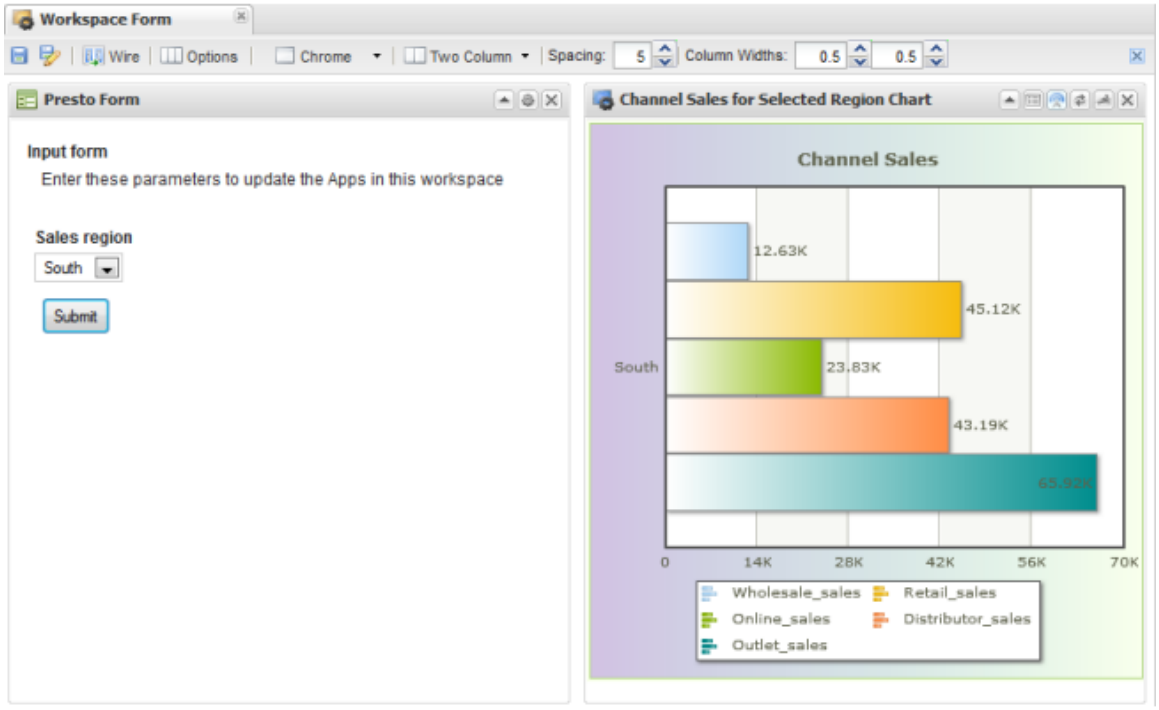
2. To move controls, simply drag and drop the control as needed.
3. To change the properties of an individual control:
   a. Right-click anywhere in the control.
   b. Select **Edit**.
   c. Change the properties as needed and click **Apply**.

   See "Add a Filter for Apps in a Workspace" on page 1161 for information on control properties.
4. To delete a control, right-click anywhere in the control and select **Remove**.
5. Click **Done** once your changes are complete.

Add an Input Form for All Apps In a Workspace

For workspaces with two or more apps that have input parameters, you can consolidate all of the input into one form that you define in the workspace and then wire updates from this form to the other apps in the workspace.

The following example has a form in the left top cell of the workspace where users can select a sales region. This input parameter is wired to the charts in the workspace to display store and sales data for the selected region:



Note: Forms you add to a workspace only provide input to other apps in the workspace. They cannot save or update data.

In releases prior to 3.5, workspace forms were generated and saved as apps named MashZone NextGen Form. For 3.5 or later, forms are saved as configuration in the workspace.

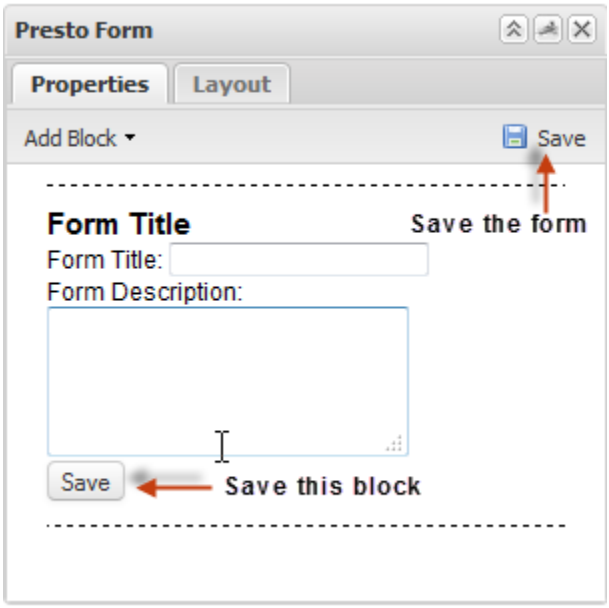
If desired, you can remove the apps for workspace forms created in previous releases. Edit the workspaces, delete the existing form, recreate it and save the workspace. Then find and delete the MashZone NextGen Form apps.

To add an input form to a workspace

1. Hide the individual input forms for each app that you are going to wire to this combined form. See "Hiding Input Forms for Apps in a Workspace" on page 1181 for instructions.

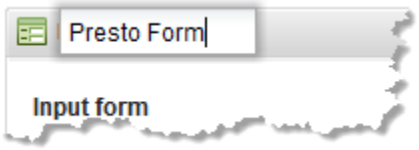
2. Open the **Palette** tab and, if needed, expand the Utilities section.
3. Drag the **MashZone NextGen Form** gadget into the workspace.

   Forms have a title block and any number additional blocks that each define one field in this form.



4. Complete the **Form Title** block. See "Title Block" on page 1168 for details.

   If you don't need a title and description or instructions, you can leave this block empty and save it.
5. Add fields to the form as needed. To add a field:
   a. Click **Add Block**.
   b. Choose the type of field to add.
   c. Complete the properties for each block. For details, see:
      - "Input Block" on page 1169
      - "Text Area Block" on page 1169
      - "Select Block" on page 1169
      - "Radio Block" on page 1169
   d. Click **Save**, in the bottom right corner, to save this block.
6. When you have finished adding fields to the form, click **Save** in the upper right corner to save this form.
7. If desired, update the title shown in the title bar for the input form. Double-click *MashZone NextGen Form* in the title bar and enter the title you want:



8. Add wiring to connect these input parameters to the input parameters or other properties for other apps in this workspace.

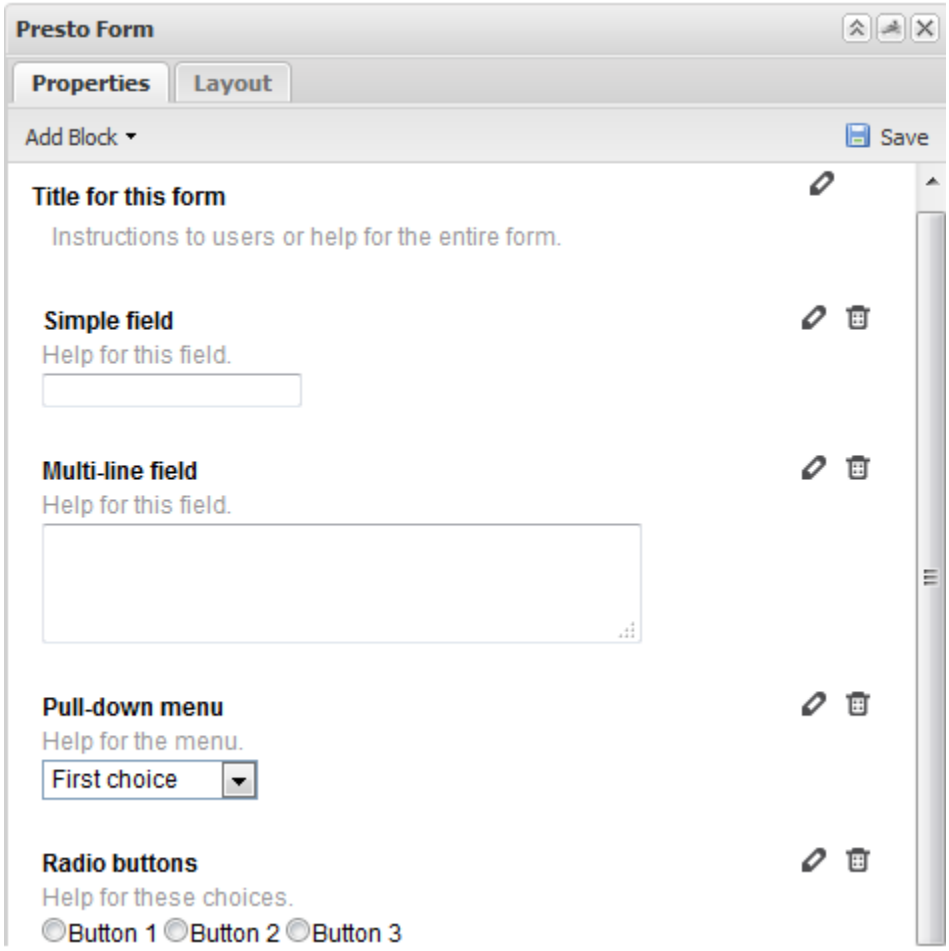
   You should wire both of the two topics that workspace forms publish to ensure that all changes to input parameters are handled:

   - `Form Submit`
   - `Form Change`

   See "Wiring App Interactions in a Workspace" on page 1175 for basic instructions on adding wiring to a workspace.

Edit a Workspace Form

To edit an input form that you have added to a workspace, find and open the workspace in Mashboard (see "Find Views, Apps or Workspaces in Mashboard" on page 1123 for instructions). Click **Configure** to begin editing the form.



Use the **Properties** tab to update fields and labels in the form. Click **Edit** for a specific block in the form and update block properties as needed. For details on block properties, see:

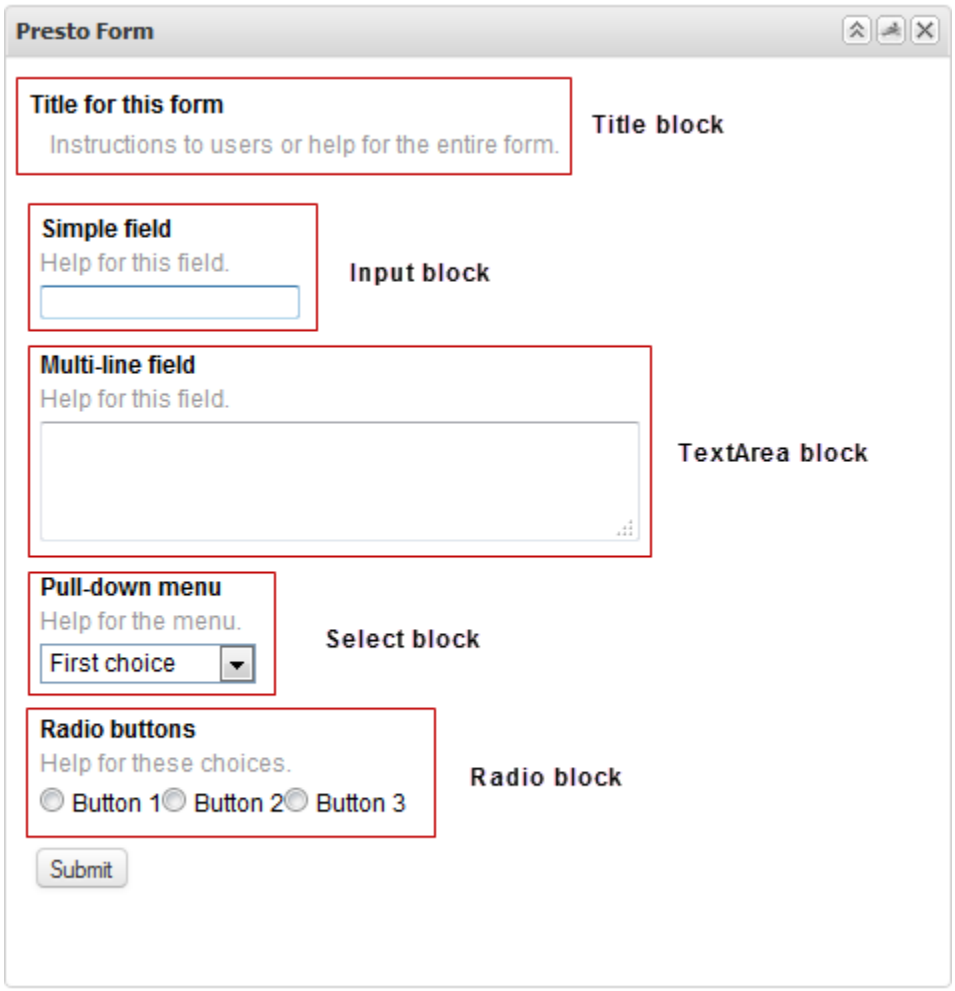
- "Title Block" on page 1168
- "Input Block" on page 1169
- "Text Area Block" on page 1169
- "Select Block" on page 1169
- "Radio Block" on page 1169

You can use **Delete** to remove fields from the form. To rearrange fields, delete fields and add them back again with **Add Block** in the order you want them.

The **Layout** tab allows you to update the **Height** for the form, chrome properties and properties for the toolbar buttons to include in the form's title bar. See "Control Title Bars and Borders" on page 1128 for more information on these properties.

Workspace Form Fields

The title and each field in a workspace form is defined as a *block*. There are five types of blocks you may use in a workspace form, shown below:



The form has a **Submit** button which publishes a `Form Submit` topic containing the values of all the form fields. Forms also have a `Form Change` topic

Both published topics can be wired to the input parameters for other apps in the workspace.

Title Block

The title block provides a title for the form (**Form Title**) and instructions or other helpful information for users (**Description**). Both fields are optional. However, you cannot delete this block.

Input Block

This block defines a simple field where users enter the value for this input parameter. Input fields have these properties:

- **Name** = this is the internal property name for this parameter that appears in wiring and in the published topics for this form.
- **Title** = the label in the form for this field.
- **Description** = helpful hints to users about this field.

Text Area Block

This block defines a multi-line field where users can type longer values, messages or descriptions. Text area fields have these properties:

- **Name** = this is the internal property name for this parameter that appears in wiring and in the published topics for this form.
- **Title** = the label in the form for this field.
- **Description** = helpful hints to users about this field.

Select Block

This block defines a pull-down menu where users select one value for this input parameter from a list of values that you define. Select fields have these properties:

- **Name** = this is the internal property name for this parameter that appears in wiring and in the published topics for this form.
- **Title** = the label in the form for this field.
- **Description** = helpful hints to users about this field.
- An Options section where you define the values that users can choose. Each option has a label and a value property. Once you enter an option, another empty pair of fields opens. Complete the options and then save the block.
- **Label** = the label that users see and select for one option.
- **Value** = the actual value that should be assigned to this input parameter if users select this option.

Radio Block

This block defines a list of options that users can choose from as individual radio buttons. Users can select only one radio button to choose the option for this input parameter. Radio blocks have these properties:

- **Name** = this is the internal property name for this parameter that appears in wiring and in the published topics for this form.

- **Title** = the label in the form for this field.
- **Description** = helpful hints to users about this field.
- An Options section where you define the individual radio buttons. Each option has a label and a value field. Once you enter an option, another pair of empty fields opens for a new radio button. Complete the options and then save the block.
- **Label** = the label for this radio button.
- **Value** = the actual value that should be assigned to this input parameter if users select this radio button.

Add Google Gadgets to a Workspace

To use Google gadgets in workspaces along with MashZone NextGen apps

1. Get the <script> tag to use for the gadget you want to add to a workspace.
2. Click the **Palette** menu and expand **Utilities**.
3. Drag **Google Gadget** into the workspace and drop it into the position you want it.
4. Complete these properties:
   - **Gadget Title** = change the title for this gadget.
   - **Script Tag** = paste the code you copied for the Google gadget here.
   - Set the **Show Title** option if desired.
5. If available, set **Layout** options for the gadget:
   - **Height** = the height in pixels for this gadget.
   - **Chrome Options**: set one of the options to control the title bar and border for this gadget:
     - **Chrome** = set this option to display both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
     - **No Header** = set this option to hide the title bar, but display a border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
     - **No Chrome** = set this option to hide both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
   - **Closeable** = clear this option to hide the Close button in the toolbar.
   - **Draggable** = clear this option to keep this gadget always in its current position in the workspace.

- **Row Span** = if this layout has multiple rows, set the number of rows this gadget should fill.
- **Column Span** = if this layout has multiple columns, set the number of columns this gadget should fill.

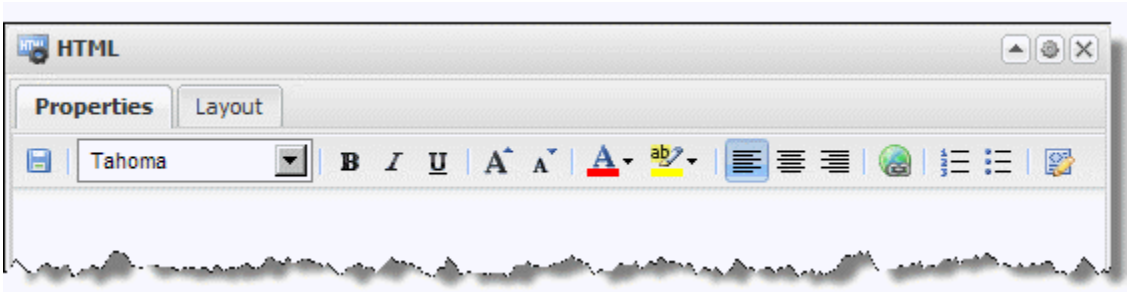
6. Click **Apply**.

Add an HTML Widget to a Workspace

To add fragments of well-formed HTML as a widget in a workspace.

1. Click the **Palette** menu and expand **Utilities**.
2. Drag **HTML** into the workspace and drop it where you want it in the workspace.

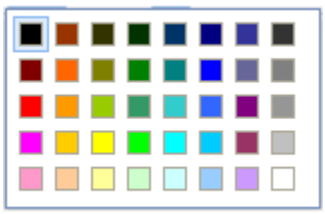
   The widget opens with a canvas area where you can enter text and use basic formatting tools, paste HTML code, or edit the HTML source code.



   The HTML editor opens in preview mode where you can enter text and apply formats without seeing the HTML code. The toolbar buttons provide familiar editing capabilities. Fonts are limited to:

   - Arial
   - Courier
   - Tahoma (default)
   - Times New Roman
   - Verdana

   And colors to:



   Use the toolbar to switch the view and edit the source HTML.

3. If available, set **Layout** options for the gadget:
   - **Height** = the height in pixels for this gadget.
   - **Chrome Options**: set one of the options to control the title bar and border for this gadget:
     - **Chrome** = set this option to display both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
     - **No Header** = set this option to hide the title bar, but display a border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
     - **No Chrome** = set this option to hide both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
   - **Closeable** = clear this option to hide the Close button in the toolbar.
   - **Draggable** = clear this option to keep this gadget always in its current position in the workspace.
   - **Row Span** = if this layout has multiple rows, set the number of rows this gadget should fill.
   - **Column Span** = if this layout has multiple columns, set the number of columns this gadget should fill.
4. Click **Apply** to save any layout changes.
5. Click **Save** in the HTML Gadget toolbar to save your content and HTML code.

Add Other Media as Objects to a Workspace

To add other types of media as gadgets to a workspace if they use the HTML <object> tag.

1. Click the **Palette** menu and expand **Utilities**.
2. Drag **Object** into the workspace and drop it into the position you want it.
3. Complete these properties:
   - **Title** = change the title for this gadget.
   - **Object Tag** = paste the <object> tag with the appropriate code needed for this media file.
4. If available, set **Layout** options for the gadget:
   - **Height** = the height in pixels for this gadget.
   - **Chrome Options**: set one of the options to control the title bar and border for this gadget:

    - **Chrome** = set this option to display both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
    - **No Header** = set this option to hide the title bar, but display a border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
    - **No Chrome** = set this option to hide both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
- **Closeable** = clear this option to hide the Close button in the toolbar.
- **Draggable** = clear this option to keep this gadget always in its current position in the workspace.
- **Row Span** = if this layout has multiple rows, set the number of rows this gadget should fill.
- **Column Span** = if this layout has multiple columns, set the number of columns this gadget should fill.

5. Click **Apply**.

Add Web Pages to a Workspace

To pull in any accessible web page (external or internal) as a gadget in a workspace

1. Click the **Palette** menu and expand **Utilities**.
2. Drag **Web Page** into the workspace and drop it into the position you want it.
3. Complete these properties:
    - **Page Title** = change the title for this gadget.
    - **Page URL** = Enter the URL for the web page to pull in.
    - Set the **Show Scrollbar** option if the page content is likely to be larger than the space within the workspace for this gadget.
    - Clear the **Show Forward/Backward Buttons** if you want to prevent users from returning to previous pages.
4. If available, set **Layout** options for the gadget:
    - **Height** = the height in pixels for this gadget.
    - **Chrome Options**: set one of the options to control the title bar and border for this gadget:
        - **Chrome** = set this option to display both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.

- **No Header** = set this option to hide the title bar, but display a border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
- **No Chrome** = set this option to hide both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.

- **Closeable** = clear this option to hide the Close button in the toolbar.
- **Draggable** = clear this option to keep this gadget always in its current position in the workspace.
- **Row Span** = if this layout has multiple rows, set the number of rows this gadget should fill.
- **Column Span** = if this layout has multiple columns, set the number of columns this gadget should fill.

5. Click **Apply**.

Add Gadgets With JavaScript to a Workspace

Some gadgets and widgets use <script> tags in HTML which you can use to add these gadgets to a workspace.

To add a javascript gadget to the workspace

1. Click the **Palette** menu and expand **Utilities**.
2. Drag **Script Embed** into the workspace and drop it into the position you want it.
3. Complete these properties:
   - **Script Title** = change the title for this gadget.
   - **Script Tags** = paste any <script> tags needed for this gadget here.
   - **Inline** = set this property to add the <script> tag directly to the workspace rather than adding the gadget securely in an iFrame.

     This option is clear by default which isolates the gadget from styles or code used in other apps or gadgets in the workspace.
4. If available, set **Layout** options for the gadget:
   - **Height** = the height in pixels for this gadget.
   - **Chrome Options**: set one of the options to control the title bar and border for this gadget:
     - **Chrome** = set this option to display both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.

- **No Header** = set this option to hide the title bar, but display a border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.
- **No Chrome** = set this option to hide both the title bar and the border around this gadget. The title bar also contains the toolbar which allows users to edit inputs or other properties or to minimize/maximize the gadget.

- **Closeable** = clear this option to hide the Close button in the toolbar.
- **Draggable** = clear this option to keep this gadget always in its current position in the workspace.
- **Row Span** = if this layout has multiple rows, set the number of rows this gadget should fill.
- **Column Span** = if this layout has multiple columns, set the number of columns this gadget should fill.

5. Click **Apply**.

Wiring App Interactions in a Workspace

Wiring interactions allows the apps in a workspace to react to events in other apps in the same workspace. This interaction, also known as *inter-app communication*, is based on *publishers* and *subscribers*.

Note: In some specific situations, apps may also automatically have events wired.

Apps may publish information about events that occur to them, such as a user clicking a row in a grid or entering input parameters in a form. Publishing an event sends a message with the event *payload* to other, interested apps in the workspace.

Apps define their interest in receiving event updates by subscribing to the event, known as a *topic*. Subscriptions define which apps in the workspace receive messages for events when they occur. Subscribing apps use the payload information to update their own data, change selection highlighting or perform some other appropriate action.

For more information on published and subscription events, see "Events and Topics for Wiring Basic and Custom Apps" on page 1178.

To wire interactions

1. Find and open a workspace or start a new workspace with at least two apps that should be wired to work together.
   - If the apps in a workspace have input parameters, you can create a form to allow users to update inputs for all the apps in one place. See "Add an Input Form for All Apps In a Workspace" on page 1164 for more information.

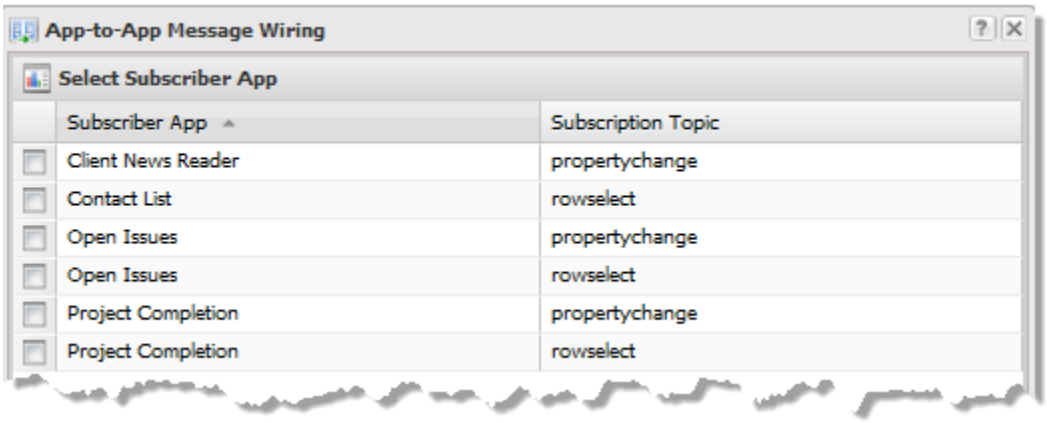
- You can also permanently suppress the individual input forms for the apps that w ly parameters. See "Hiding Input Forms for Apps in a W 81" for details.

2. Click **Wire**.

   This opens the App Wiring wizard.

3. Click **Create New Wiring**.

   The App Wiring wizard lists the apps in this workspace that have defined topics they can subscribe to along with their name for the subscription.



   Basic apps generally have two subscription topics:

   - `propertychange` subscription that uses the published message to update input parameters for the app and refresh its data.

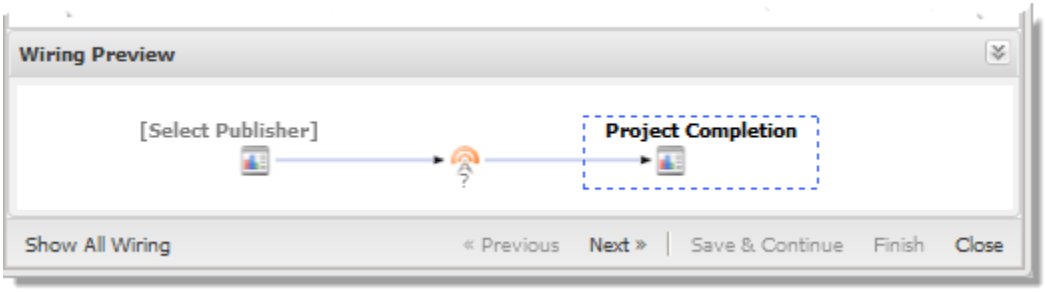
     If the app contains a chart view, this subscription can also contain user-defined properties that are defined for the chart. See "Include Dynamic Content in Captions and Labels" on page 906 for more information.

   - `rowselect` subscription for their associated DataTable that updates what is shown as selected in the app.

   See "Subscription Topics for Basic Apps" on page 1179 for more information on the events for these subscription topics.

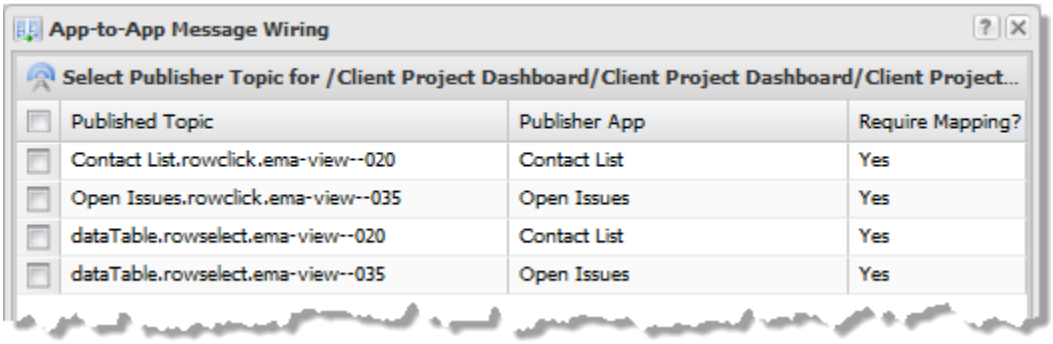
4. Check the box for the appropriate topic for the app or DataTable that should receive updates from another app.

   This is the subscriber which now shows in the wiring preview:



5. Click **Next**.

   The App Wiring wizard lists the apps or associated DataTables that have defined topics they can publish along with the name of the published topic. See "Publish Topic Messages for Events in Basic Apps" on page 1179 for more information.

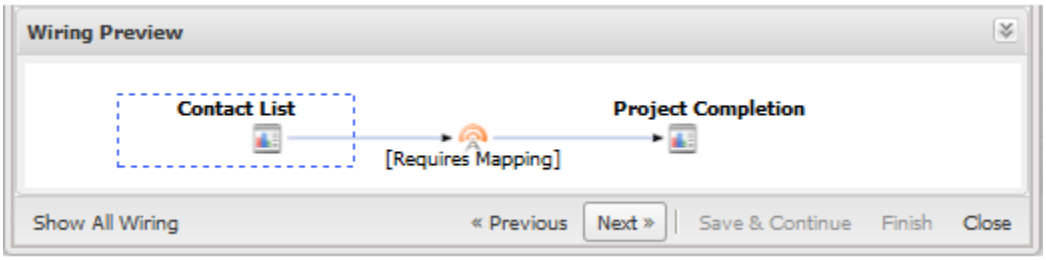


Tip: The events that charts or some other views publish are not always immediately visible for wiring. For example, the `Cell-Click` event is not available in apps with a grid of data until you click in at least one row of the grid.

If no publisher information displays or the publish event you want to use is not listed, close the App Wiring wizard and try clicking appropriate areas in the apps or performing other tasks that should publish events. Then try wiring again.

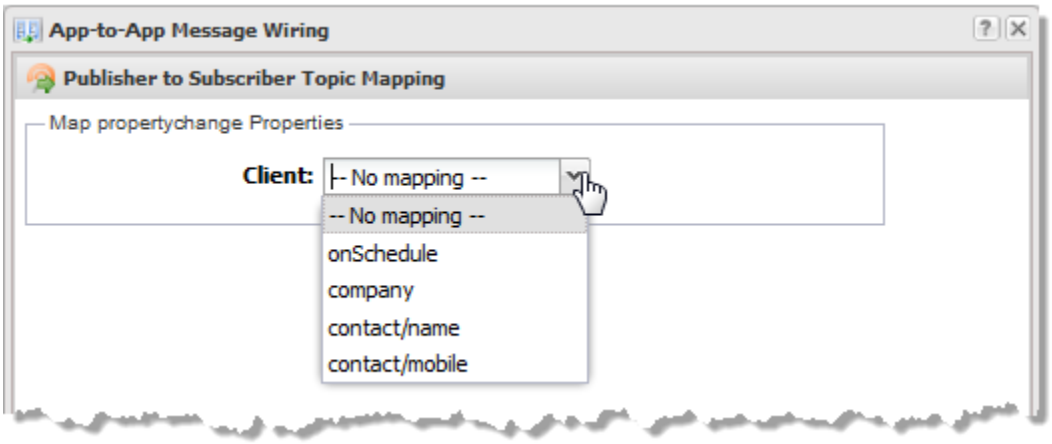
6. Check the box for the app or DataTable that should send updates to the subscriber app.

   This is the publisher which now shows in the wiring preview:



7. Click **Next**.
8. If mapping is required, select the appropriate publish topic fields that should be mapped to subscription topic fields.

> **Note:** If the publish topic field you want to map does not appear in the list of available fields for mapping, you can return and select the `datatable.rowselect` publish topic associated with the publish app. This topic includes all fields for the mashable or mashup results, not just those that are included in the view.



9. Click **Finish** to save this wiring and close the App Wiring wizard or click **Save and Continue** to save this wiring and continue wiring apps.

Events and Topics for Wiring Basic and Custom Apps

Custom apps can optionally define their own set of events that they publish topics for or subscribe to. If the custom app supports loose coupling, the topics for these events are declared in the App Spec and are visible in Mashboard when you add wiring to a workspace app. MashZone NextGen developers should see "Declare App Topics and Payloads" on page 1285 for more information.

Basic apps, created in App Maker or created for views added to a workspace in Mashboard, use built-in or pluggable views that can have both standard and view-

specific events. Basic apps publish topics and can subscribe to topics for both standard and view-specific events.

Publish Topic Messages for Events in Basic Apps

Basic apps publish messages to topics for events that are supported by the views included in the app (*view-specific events*) or by the *DataTable* containing the results from the mashable or mashups:

- **View-Specific Events:**
  - For built-in MashZone NextGen views, see documentation for the specific view to find the topics published for events in that view. For links, see "Views for Mashups and Mashables" on page 895.
  - For pluggable views, the view must register the topics that it publishes for view-specific events with MashZone NextGen. Information on view-specific events may be available in help for that view.

    MashZone NextGen developers should see "Pluggable View Classes: Triggering and Handling Events" on page 1059 for more information.
- **DataTable Events:**

  DataTables, with the results for the mashable or mashup, have a single `rowselect` event which the app for that DataTable publishes as a `rowselect` topic. All built-in MashZone NextGen views that support some type of click event also trigger a `rowselect` event in the DataTable. See online help for information on the events that publish topics in MashZone NextGen built-in views.

  Pluggable views can also optionally choose to trigger DataTable `rowselect` events. MashZone NextGen developers should see "Pluggable View Classes: Triggering and Handling Events" on page 1059 for more information.

  MashZone NextGen developers should see "About DataTable Events" on page 1527 for more information on the interaction of DataTable events, views and apps.

Thus a user action can trigger both a view-specific event, such as `click`, and the `rowselect` event for the associated DataTable. The message payload for view-specific events typically includes *only* those fields (columns) that are included in the view. The payload for `rowselect`, however, includes *all* fields (columns) in the DataTable for that row.

Subscription Topics for Basic Apps

There are two standard subscription topics that basic apps can subscribe to:

- `propertychange` = this topic is available if the basic app has input parameters. Basic apps for views added directly to a workspace have input parameters if the mashable or mashup does.

Note: If the basic app includes a chart view, this subscription topic can also contain user-defined properties used in the view. See "Include Dynamic Content in Captions and Labels" on page 906 for more information.

Messages published to this subscription topic update input parameters for the app which refreshes the app's data. If the topic includes user-defined properties for a chart view, published messages also update the dynamic content in chart captions or labels. You map the published message payload to these input parameters and user-defined properties during wiring.

- `rowselect` = this topic is available for the *DataTable* with the mashable or mashup results for the subscribing app.

  Messages from the `rowselect` subscription topic update which row of data in mashable or mashup results is currently flagged as selected in the app's DataTable. This in turn generally updates the views in the app, allowing them to change their visual properties, if that is appropriate. A grid, for example, may highlight a row to show it is selected or clear highlighting to show deselection.

  MashZone NextGen developers should see "About DataTable Events" on page 1527 for more information on the interaction of DataTable events, views and apps.

  Generally, MashZone NextGen built-in views with some type of 'click' event also listen for row selection updates in the DataTable. This allows synchronization of row selection across built-in and pluggable views when multiple views are included in one basic app, but also supports synchronization based on subscription messages from other apps.

  For example, a user selects one row in a grid. This automatically causes the related slice in a pie to slide out or the related bar in a bar chart to be highlighted.

  Pluggable views can also optionally choose to listen for `rowselect` events in the DataTable.

  MashZone NextGen developers should see "Pluggable View Classes: Triggering and Handling Events" on page 1059 for more information.

  Note: If you wire a DataTable `rowselect` published topic from one app to a Datable `rowselect` subscription in another app in a workspace, you can mimic the automatic row selection behavior of a basic app with multiple views, with *some known limitations*. Mapping defines the relationship between the rows of results for the two apps.

  Known limitations include:

  - When the DataTable for the subscribing app is paginated, matches to the row identified in a published `rowselect` message may be missed if they are not in the current page of results in the DataTable. In this case, nothing happens to the views in subscribing app, making it look like there is no matching row.

- If there are multiple rows that match a published selection, only the first match will be affected because the DataTable only tracks one selected row.

MashZone NextGen developers should see "About DataTable Events" on page 1527 for more information.

Hiding Input Forms for Apps in a Workspace

By default, workspaces hide the input form for each app within the workspace, with *one* exception: when the app is first loaded in the workspace. This allows users to set input parameters initially so that each app can retrieve and display information. Users can open an app's input form at any later time with the **App Tools** button in the app title bar.

When you wire interactions between apps in a workspace, information published in events can be used to supply input parameters to the subscribing apps instead of having users enter this information. Or you may create a combined input parameter form that supplies inputs for all apps in the workspace in one, single form so that users do not have to fill in and apply several forms to see all the information for the workspace.

In these cases, users should *not* be entering values for input parameters in individual apps. You can have the workspace completely hide the input form for individual apps.

To hide input forms

1. Find and open the workspace in Mashboard. See "Find Views, Apps or Workspaces in Mashboard" on page 1123 for instructions.
2. Click **Configure** in the title bar for an app with an input form that you want to hide.
3. Open the Inputs tab and set the **Hide Inputs** option.

Synchronizing Wired Apps When a Workspace Loads

Wired interactions between apps in a workspace generally depend on a user action. In some cases, however, you may want the apps in the workspace to be consistent when the workspace first loads -- *before* a user had done anything.

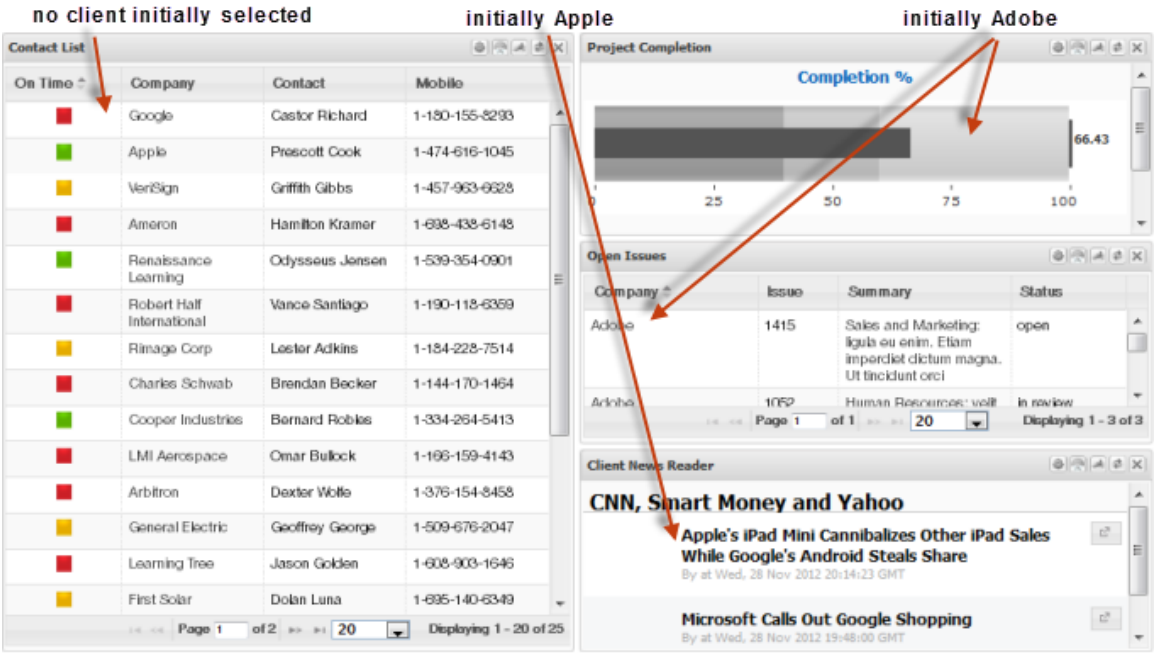
You can synchronize some interactions when a workspace loads using the DataTable `rowselect` publish topic. See "Events and Topics for Wiring Basic and Custom Apps" on page 1178 for more information about this event and the published payload.

If wiring for a workspace includes subscriptions to any DataTable `rowselect` event, the workspace will automatically trigger `rowselect` for the first row in the DataTable when the app associated with that DataTable is rendered. This publishes a `rowselect` message and subscribing apps use this message to reflect that event. The two most common applications of this are: "Update Input Parameters on Workspace Load" on page 1182 and "Synchronize Row Selection on Workspace Load" on page 1184

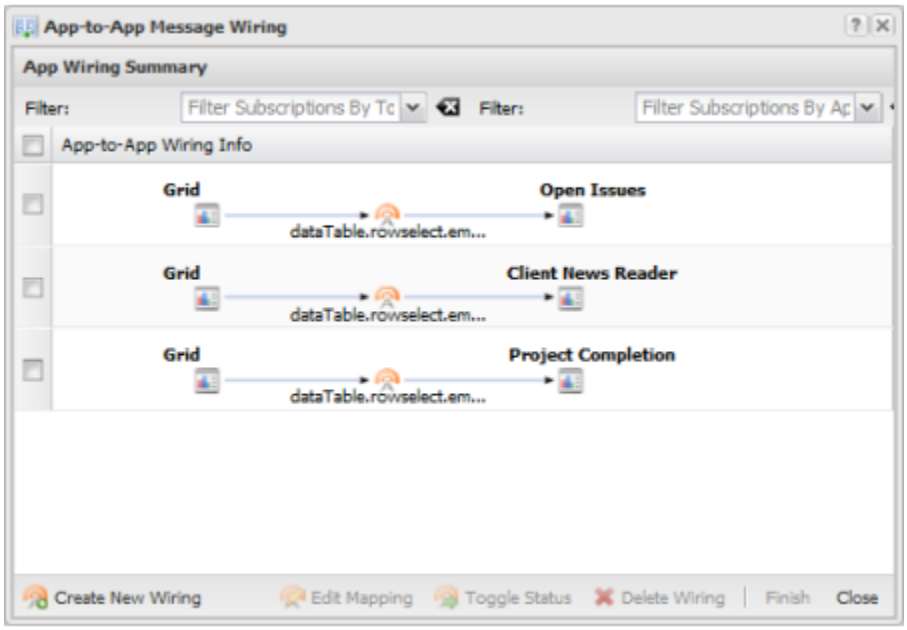
Update Input Parameters on Workspace Load

In this case, the selection from one app updates input parameters to subscribing apps allowing them to refresh their data. The workspace created in the MashZone NextGen Quick Tour videos is an example.

This workspace has a user select a project from a grid of current clients. Once a client is selected, updated completion status, open issues and financial news for that client displays in the remaining apps in the workspace. When the workspace first opens, however, no client is selected and the data in the other three apps is not consistent:

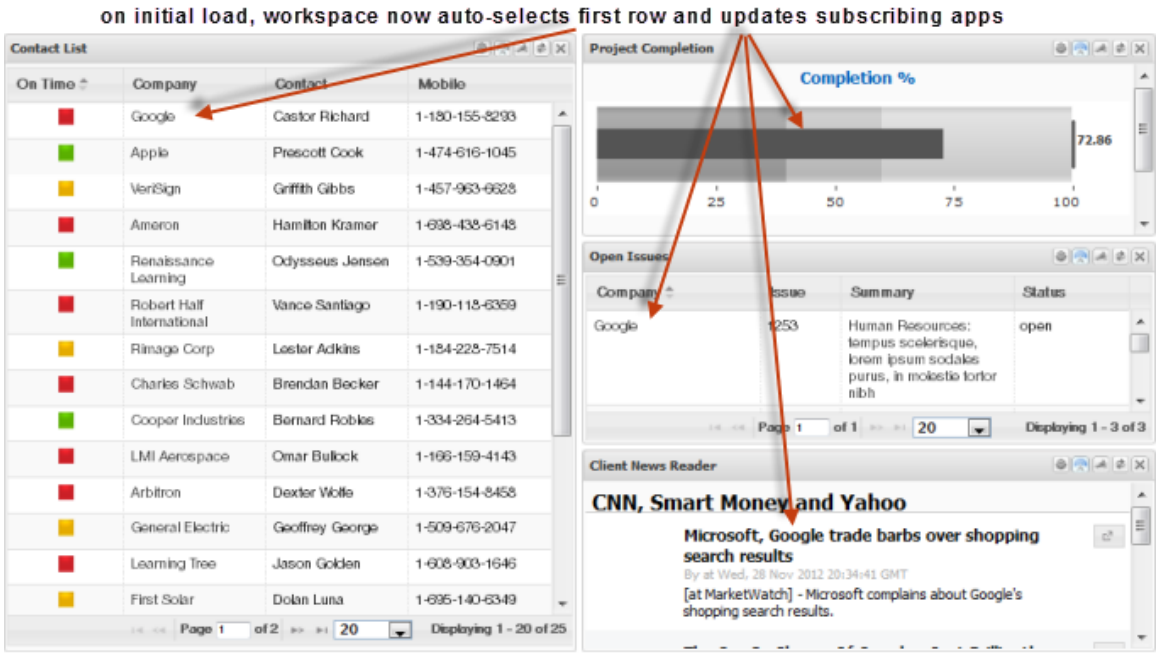


To synchronize the workspace when it first loads, the subscribing apps must wire their `propertychange` topic to the publish `rowselect` topic for the DataTable of the app that should trigger updates. For the Quick Tour example, the wiring looks like this:



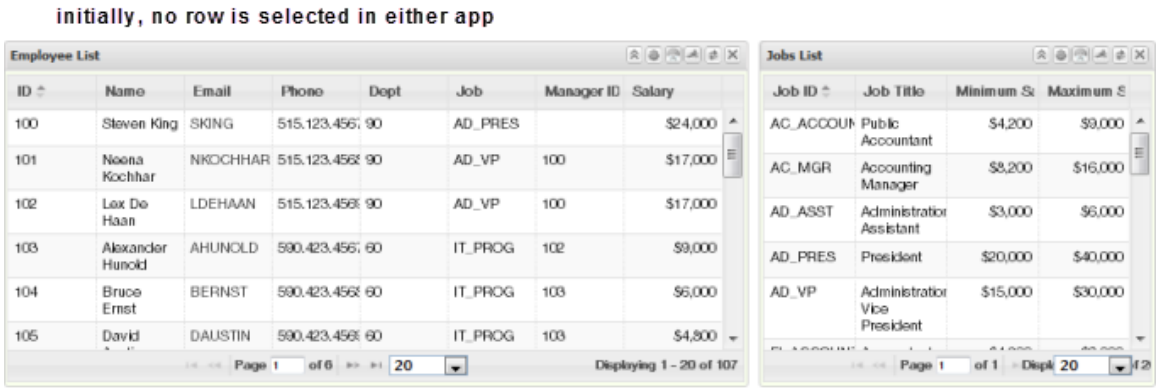
The Contact List with a Grid view is the app that users select a client project from. The DataTable for Contact List becomes the publisher that is wired to the `propertychange` topic of the Project Completion, Open Issues and Client New Reader apps.

After this wiring is in place, when the workspace loads the data consistently reflects information for the first client listed in Contact List:



Synchronize Row Selection on Workspace Load

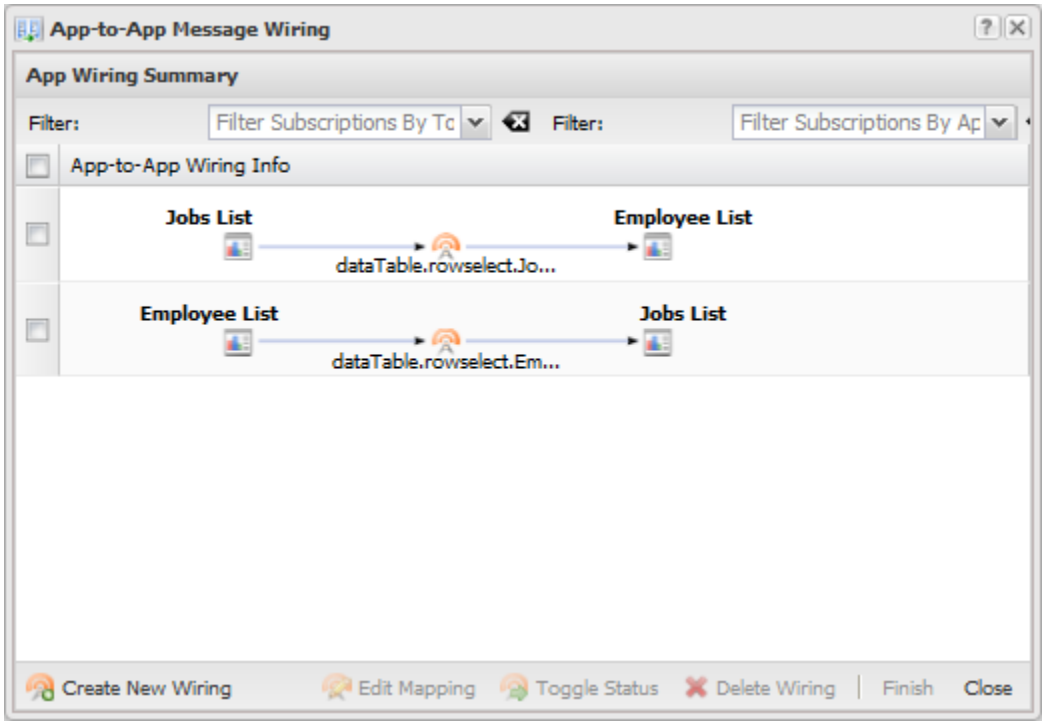
In this case, the workspace needs to indicate a relationship between one row in an app and a corresponding row in another app. This example relates a list of employees with a list of job titles:



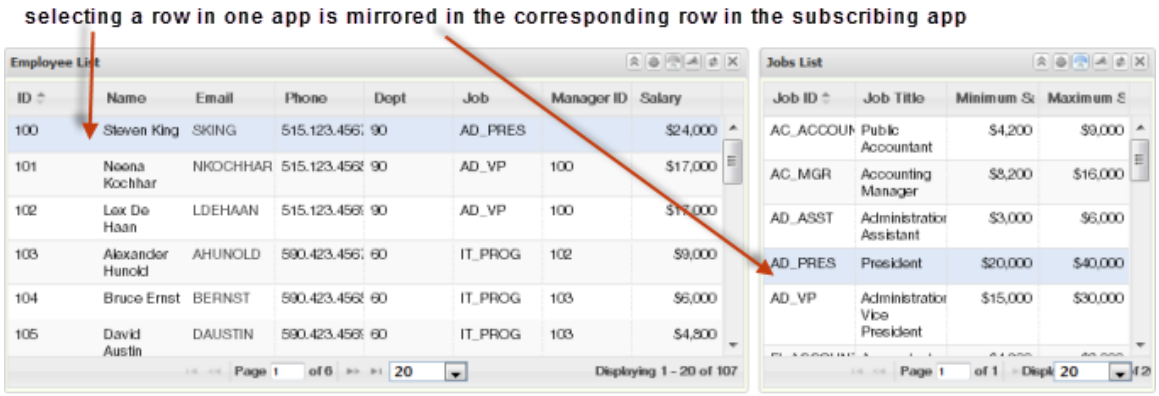
Wiring the publish `rowselect` topic for the DataTable of one app to the subscribing `rowselect` topic for the DataTable of another app allows the subscriber app to mirror row selection in the publisher.

Note: This wiring has some specific limitations. If the apps are paginated, each DataTable contains just a single 'page' of rows in the dataset. This can result in matching errors, incorrectly showing no match because the matching row is not present in the current page.

Mapping published topic fields defines the relationship between the two DataTables.



Once wiring is complete, selection is synchronized:



Manage App Interaction Wiring in Workspaces

Common editing and management tasks for workspace wiring include:

- Toggle Wiring Subscriptions Off or On
- Edit Subscription Property Mappings
- Hiding Input Forms for Apps in a Workspace

Toggle Wiring Subscriptions Off or On

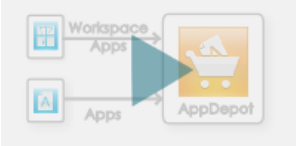
In the toolbar for the subscribing app, click **App Wired**. To turn off a subscription, clear the option for that subscription. To turn a subscription back on, set the option for that subscription.

Edit Subscription Property Mappings

To update [illegible]es in an existing wiring

1. Click [illegible] **Wire**.
2. Select the subscription you want to edit and click **Edit Mapping**.
3. Change the property mappings as needed and click **Finish** to save these updates.

Working in the MashZone NextGen Enterprise AppDepot



The MashZone NextGen Enterprise AppDepot let's you find and work with apps in your desktop browser that you or other users in your organization have created. Apps work with information and documents in your organization as well as external information.

Note: For mobile apps, see "Working in MashZone NextGen Mobile Apps" on page 1189 for more information.

Click AppDepot in the MashZone NextGen Hub main menu to open the AppDepot.

You can easily search through desktop apps in the AppDepot and review app information to help you decide whether an app may be useful.

You can open and work with any number of apps that you find. Or save apps to your My Favorties page to quickly find and open your favorite apps.

You can also create personal copies of apps, add them to My Favorites and tweak their properties to fit just what you need to see or work with.

You can rate or comment on apps. And you can share apps with other users or open apps other users have shared with you. See "AppDepot Tabs and Toolbars" on page 1187 for more information.

If you have permission to create apps, See "Create a Basic App" on page 1093.MashZone NextGen administrators should also see "Managing Pending Apps in

the AppDepot" on page 1187 and "Grant Run Permissions to Published Apps in the AppDepot" on page 107.

AppDepot Tabs and Toolbars

- **My Favorites**: contains desktop apps that you have added to your favorites. Use My Favoriates to quickly find and open your favorite apps.
- **All Apps**: find any desktop app in the AppDepot from this tab. Review information on desktop apps or open them to begin working with them. You can also add desktop apps to your My Favorites page.
- *Pending Apps*: a tab only AppDepot managers can see and work with. This tab lets MashZone NextGen administrators review and accept or decline new apps or updates to apps that you or other users have submitted to the AppDepot.
- Add to My Favorites
- Review, comment, rate and other App functions
- Open App (in a new window or tab)
- Embed App
- Publish App
- Share App

Managing Pending Apps in the AppDepot

MashZone NextGen users and developers who create apps in MashZone NextGen Hub publish these apps to the AppDepot to make them available to other users in the AppDepot, for desktop access, or in the MashZone NextGen Mobile apps, for access in mobile phones or tablets.

Publishing is a managed process to ensure that apps meet requirements and are deployed and updated with minimal disruption to other users. MashZone NextGen administrators act as the AppDepot manager, the gatekeepers that review pending apps to approve or decline them. For more information on this workflow, see "Publish Apps to the AppDepot" on page 1114.

Each organization defines the approval criteria for pending apps. Common criteria includes having sufficient information or screen shots to help users understand the app.

Note: Only MashZone NextGen administrators have permission to view, approve or decline pending apps in the AppDepot.

To manage pending apps

1. Click **Pending** in the AppDepot menu to see the list of apps that have been submitted for review.

Use **Search** or the Category, Provider and Tag filters to quickly find a specific pending app.

2. To run the app, click **Open** (in the bottom right corner).
3. Click the app name to open the AppDepot artifact page for this app and review information or the workflow history.

   Note: To ensure minimal disruption, approved and pending apps are separate copies of the original apps created in MashZone NextGen Hub and have separate artifact pages.

4. To approve this app, click **Approve** from the app information page.

   A basic checklist of the status of common criteria will be added to the notification sent to the app owner. Add optional comments and click **Send to AppDepot**.

   This adds a new app or updates the existing app in the AppDepot. If the app is mobile compatible, this also adds a new app or updates the existing app the MashZone NextGen Mobile apps. Finally, this updates the status of the original app in MashZone NextGen Hub and adds an entry to the workflow history.

   Note: Run permissions for the MashZone NextGen Hub app automatically follow to the app copy in AppDepot and the MashZone NextGen Mobile apps. You may need to update these run permissions to provide appropriate access. See "Grant Run Permissions to Published Apps in the AppDepot" on page 107 for instructions.

5. To decline this app, click **Decline** from the app information page.

   To add comments for any of the basic app checklist items, click that item. Optionally, add additional comments to the notification and click **Send Back to Producer**.

   This removes the pending copy of the app from the AppDepot and sends a notice to the app owner. This also updates the status of the original app in MashZone NextGen Hub and adds an entry to the workflow history.

About MashZone NextGen Mobile

MashZone NextGen Mobile includes:

- Features in MashZone NextGen Hub that allow you to create mobile-compatible views and apps.
- Native apps that you can install in mobile phones and tablets. MashZone NextGen Mobile apps are 'mobile editions' of the MashZone NextGen Enterprise AppDepot allowing you to find, view and work with mobile-compatible apps that are hosted in MashZone NextGen.

  See "Installing MashZone NextGen Mobile Apps in Mobile Devices" on page 1189 for instructions.

Then open MashZone NextGen Mobile in your mobile phone or tablet, login to MashZone NextGen and get started. See "Working in MashZone NextGen Mobile Apps" on page 1189 for a short video and other suggestions.

Installing MashZone NextGen Mobile Apps in Mobile Devices

You may install a MashZone NextGen Mobile native app in your mobile phone or tablet to find and work with mobile apps in MashZone NextGen. MashZone NextGen Mobile apps are available from the Apple App Store for iPhones and iPads. Simply download the appropriate MashZone NextGen Mobile app for your device.

Note: Native MashZone NextGen Mobile apps are not currently available for Android devices. You can, however, use mobile MashZone NextGen apps from the browser in Android devices.

Login and Get Started in MashZone NextGen Mobile

You must log into MashZone NextGen to begin working in a MashZone NextGen Mobile app. The login initially opens showing a guest account that you can use to get familiar with MashZone NextGen Mobile.

To begin working with apps from your organization, ask your MashZone NextGen administrator for the following information:

- Username and password
- Host and port information
- Whether the connection must be secure

See "Working in MashZone NextGen Mobile Apps" on page 1189 for a video and other suggestions on working in MashZone NextGen Mobile.

Working in MashZone NextGen Mobile Apps

Mobile features are available in the MashZone NextGen Hub to create views and apps that are mobile compatible. You find mobile apps hosted in MashZone NextGen using MashZone NextGen Mobile native apps that you install in your mobile phones and tablets. See "Installing MashZone NextGen Mobile Apps in Mobile Devices" on page 1189 for more information.

To enable basic apps to work in mobile devices, you must include views that are mobile compatible and choose mobile devices as a supported destination. For workspace apps, the workspace app must use a mobile-compatible layout and must also be flagged for mobile devices. See "About Desktop and Mobile Compatibility for Apps" on page 1111 for more information.

Once an app is complete, you must publish the app to the AppDepot and an AppDepot administrator must approve it. Once approved, the mobile editions of apps are available in the MashZone NextGen Mobile apps. See "Publish Apps to the AppDepot" on page 1114 for instructions.

MashZone NextGen Mobile App Features

MashZone NextGen Mobile apps allow you to log into one or more MashZone NextGen Servers for your organization. See "Login and Get Started in MashZone NextGen Mobile" on page 1189 for more information.

Once you log into your MashZone NextGen Mobile app, you can:

- Search for mobile apps by name, description or tags.
- Look for new apps or featured apps.
- Search through app categories.
- Open and use any app which you have permission to run.
- Comment and rate apps.
- Add apps to your favorites to quickly find and use them. This also updates your favorites in the AppDepot.
- Manage connection/credential information for the MashZone NextGen Servers that you work with.

Once you open an app, you use common touch controls to change input parameters, if any, or sort and filter information.

11 Custom Apps

- Working in the App Editor 1194
- Customize a Basic App or View 1198
- Create Fully Custom Apps in the App Editor 1225
- Override Browser Caches for Updates to App Resources 1289
- The Structure of a MashZone NextGen App 1290
- Parameters for App URLs 1294
- App Specification Reference 1298
- App Packages and App Files 1318

Basic apps generally display information in simple, well-defined formats and provide simple types of interactions. In some cases, this may not be sufficient to meet your requirements. MashZone NextGen developers can handle more complex requirements with:

- *Customized views/customized basic apps*. These are views you have added to a mashable or mashup using the MashZone NextGen built-in views or pluggable views to which you choose to apply more advanced customizations. This simplifies development for complex requirements, updating the look of built-in or plug-in views and optionally adding custom interactions.

  See "Customized Basic Apps and Views" on page 1192 for more information on the types of customizations possible.

- *Fully custom apps* are apps that MashZone NextGen developers create from scratch. This is more complicated as developers must produce all the resources for the view and interactions. But fully custom apps allow you to handle more complex presentation requirements or more specific interactions that are not possible by other means.

  See "Fully Custom Apps" on page 1192 for more information.

Customized Basic Apps and Views

To customize the presentation and/or behavior for views, you add one or more views to a basic app and customize the app. This uses the App Specification, app.xml, for the basic app which you edit in the App Editor.

First, follow the basic procedure to "Customize a Basic App or View" on page 1198. Then use these techniques, as needed:

- "Customize Themes for Basic Apps and Views" on page 1200 to have two or more views share a common look and color scheme.
- "Customize the Layout for Basic Apps" on page 1207 that wrap two or more views.
- "Customize Built-In Chart Attributes" on page 1218.

Note: Additional, advanced customizations are possible for the Grid and KPI View, the Template View and almost all of the built-in chart views. Examples for advanced customization techniques are coming soon to the Software AG Tech Community for MashZone NextGen at http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto/downloads.

Fully Custom Apps

Fully custom apps let developers handle more complex presentation requirements or interactions that are not possible with basic apps, even through customization. Custom apps also allow developers to meet look and feel standards for your organization.

Note: The examples of custom apps in the topics in this section are generally designed as desktop apps. Most of the techniques shown, however, apply to desktop or mobile apps.

See "Custom Mobile App Requirements" on page 1193 for more information on developing custom mobile apps for MashZone NextGen.

In some cases, you can design and add a pluggable view instead creating a custom app. See "Creating Pluggable Views or Libraries" on page 1045 for more information.

With custom apps, you start from a minimal package in the App Editor that contains stub files for the various resources used in an app. You work with these resources, testing as you go. Once the custom app is working, you can also download this final package to check this into your source control tools if needed. See "Create Custom Apps from the Base App Package" on page 1226 for basic instructions.

For examples and specific techniques you may use to create custom apps, see:

- "Paginate Mashable or Mashup Responses" on page 1491
-
- "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502
- "Declare, Get and Set Properties in Custom Apps" on page 1227
- "App Dimensions and Resizing" on page 1236
- "Enable User-Initiated or Automatic Refreshes" on page 1243
- "Handle Exceptions" on page 1247
- "Override Browser Caches for Updates to App Resources" on page 1289
- "Wiring App Interactions" on page 1249

In addition to common resources such as HTML pages and JavaScript libraries, fully custom apps also have an App Specification, app.xml, that defines all the resources in the app, properties and other information about the app. For more information on App Specs and the elements you can use, see "App Specification Reference" on page 1298.

For additional information, see the MashZone NextGen App API Reference, "The Structure of a MashZone NextGen App" on page 1290, "App Packages and App Files" on page 1318 and "Parameters for App URLs" on page 1294.

Custom Mobile App Requirements

Most of the development requirements for custom mobile apps are the same as custom desktop apps. MashZone NextGen mobile apps are HTML5 web applications with JavaScript and CSS to make them compatible with most modern mobile devices. There are only two mobile-specific requirements for custom apps:

- Use a mobile-compatible JavaScript library, such as jQuery Mobile, to allow the custom app to recognize and respond appropriately to gestures and orientation changes.
- Add mobile-compatible flags to the App Specification for the types of devices the app supports. See "<presto-meta>" on page 1306 for more information.

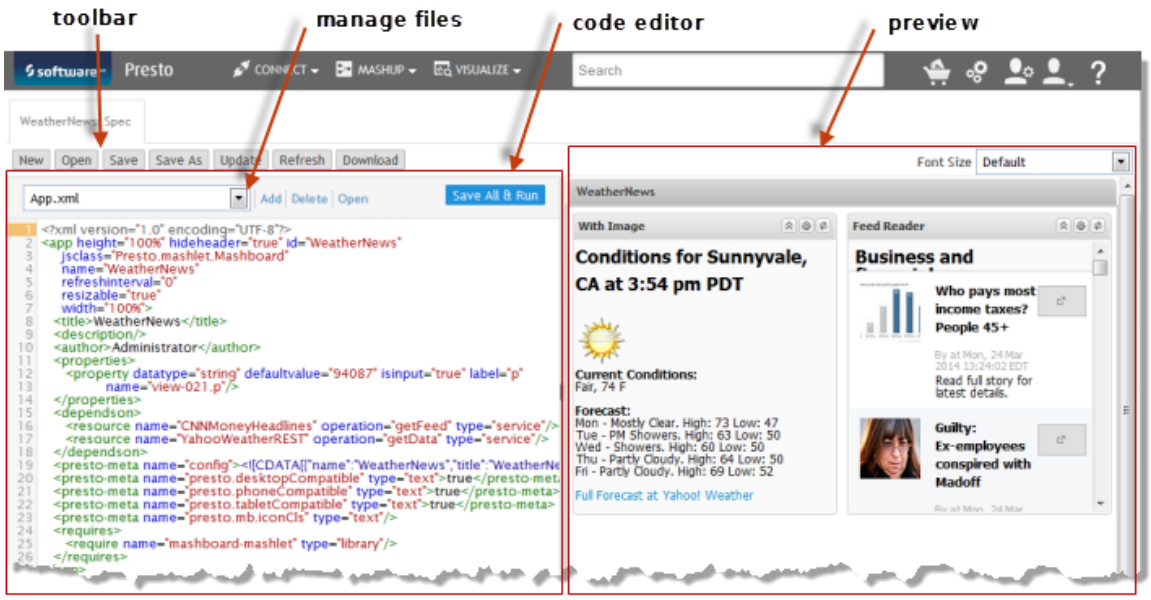
Once you have created a custom mobile app and uploaded it to MashZone NextGen, you must publish it to the AppDepot to enable mobile users to find and use it in MashZone NextGen Mobile.

Working in the App Editor

In the App Editor, you can:

- Download a new app template to begin a new custom app (see "Create Custom Apps from the Base App Package" on page 1226 for instructions).
- Upload a new app from your development environment (see "Create Custom Apps from the Base App Package" on page 1226 for instructions).
- Open custom or basic apps. See "Find and Open an App" on page 1195 for tips.
- Make simple edits and debug apps. See "Edit and Test an Open App" on page 1196 for tips.
- Download an app. See "Download an App" on page 1197 for more information.
- Update a custom app. See "Update an App" on page 1198 for more information.

The basic features of the App Editor are shown below:

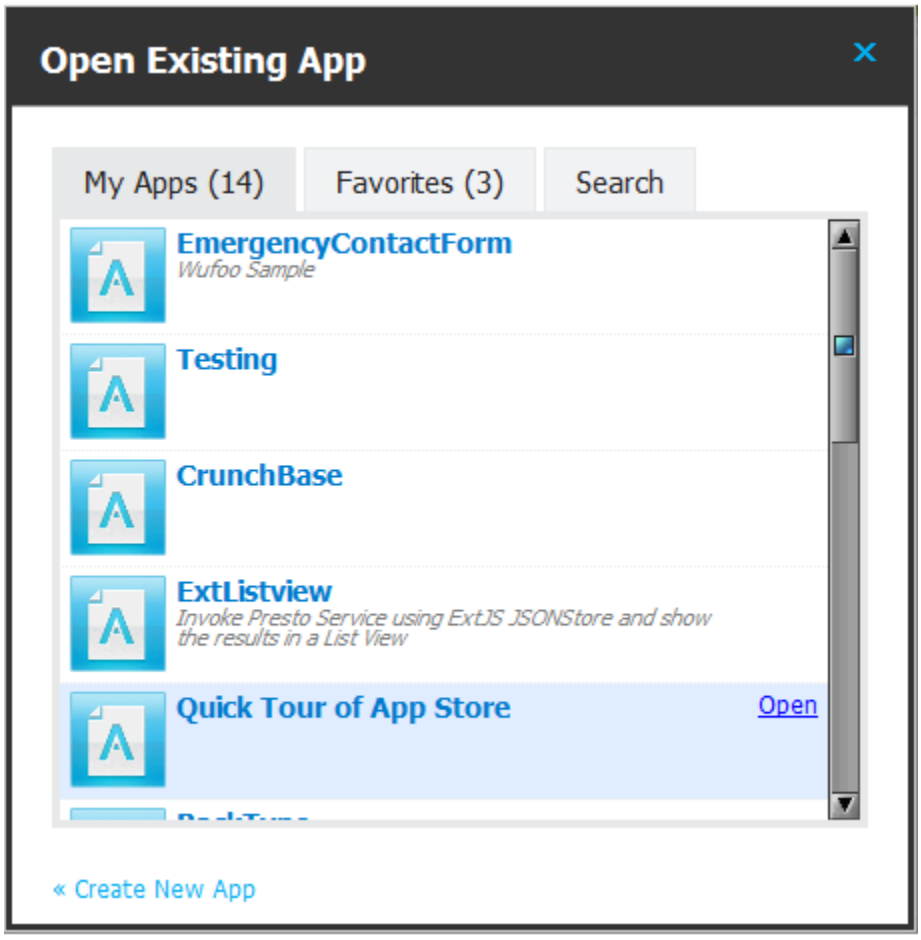


Use the *Toolbar* to manage the current app, open other apps or start a new app. Each app opens in a new tab.

You can also use **Ctrl** + **Alt** + **F** to reformat code in the Code Editor pane and the **Font Size** menu to increase the default size of code.

Find and Open an App

When you open the App Editor, the **Open App** window opens.



Use the My Apps or Favorites tabs to find apps that you created or have marked as a favorite. Or enter at least three characters of a name, description or tag in the Search tab, to find apps with a filter. Search returns the first 20 apps, so providing more characters can help narrow the results.

Click the **Open** link for a specific app to open it in theApp Editor.

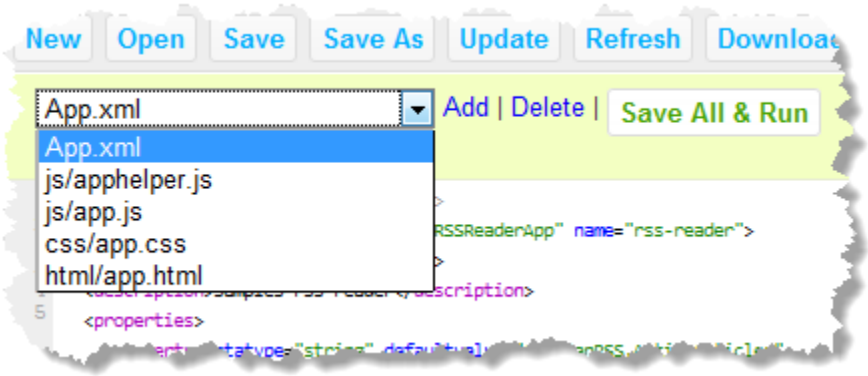
Note: You can generally see all apps. But your MashZone NextGen permissions determine which apps you can open.

You can also open an app in the App Editor from search results in Browse on the main menu.

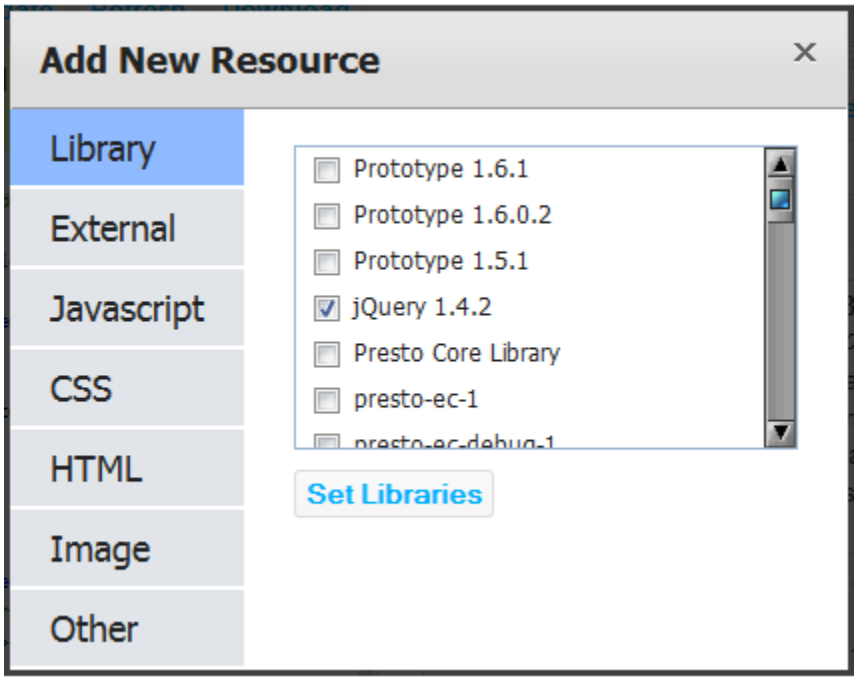
Edit and Test an Open App

Once you have an app open, the *Preview* pane renders the app when you click **Save All and Run** from the Code Editor. Use this to run and debug the app.

The *Code Editor*pane allows you to edit any text file in the app including the App Specification, HTML, JavaScript or CSS files. Use the file pull down to select which file to open.



You can delete the current file from the app with **Delete**. You can also add additional resources, either as named libraries or as files, with **Add**:



When you add or delete resources from an app, this updates the <requires> section of the App Spec:

Download an App

You can download an archive package of all resources for an app from the App Editor or from the app's artifact page. This makes it easy to update the code or resources used in the app in your own development environment.

In the App Editor	In the App Artifact Page
Find and open the app you want to download. See "Find and Open an App" on page 1195 for instructions.	Find and open the app you want to download from Search, bookmarks or other links in MashZone NextGen Hub.
Click **Download**.	Click **Manage** > **Download App**.
When prompted, save the Zip file to your development environment and extract all the app resources.	When prompted, save the Zip file to your development environment and extract all the app resources.

Once your updates are complete, upload you changes to MashZone NextGen. See "Update an App" on page 1198.

Update an App

When you update the code or resources for an existing custom app in your own development environment, you must package your updates and then upload them to MashZone NextGen to update the app. You can upload an app update package from the App Editor or from the app's artifact page.

In the App Editor	In the App Artifact Page
Archive all the app resources in a Zip file on your computer.	Archive all the app resources in a Zip file on your computer.
Find and open the app you want to update. See "Find and Open an App" on page 1195 for instructions.	Find and open the app you want to update from Search, bookmarks or other links in MashZone NextGen Hub.
Click **Update**.	Click **Manage >** **Update App**.
Click **Browse** and find the Zip file on your computer for this app . Click **Upload**.	Click **Browse** and find the Zip file on your computer for this app . Click **Upload**.

Customize a Basic App or View

In some cases, the views you create for mashables and mashups need small tweaks to meet your requirements. Rather than creating a custom app from scratch, you can start with a view or basic app and customize that to meet your needs.

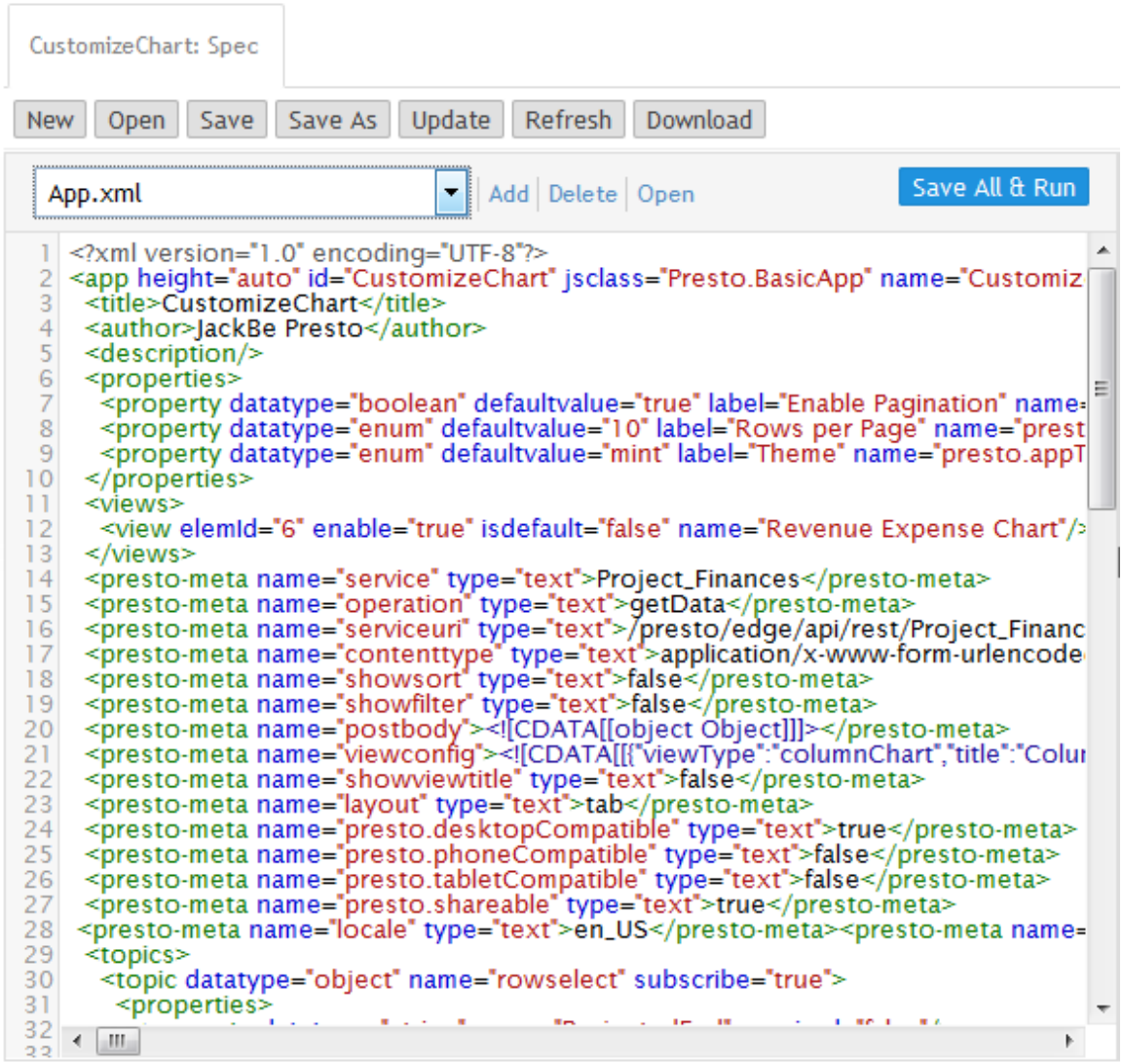
Customizations include simple visual changes, such as defining standard styles or changing the layout of views. You can also customize chart views using attributes from Fusion Charts, the underlying library for MashZone NextGen built-in chart and gauge views (not including real-time views).

More advanced customizations that use JavaScript are possible. Information and samples of advanced customizations is coming soon in the MashZone NextGen Technical Community at http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto.

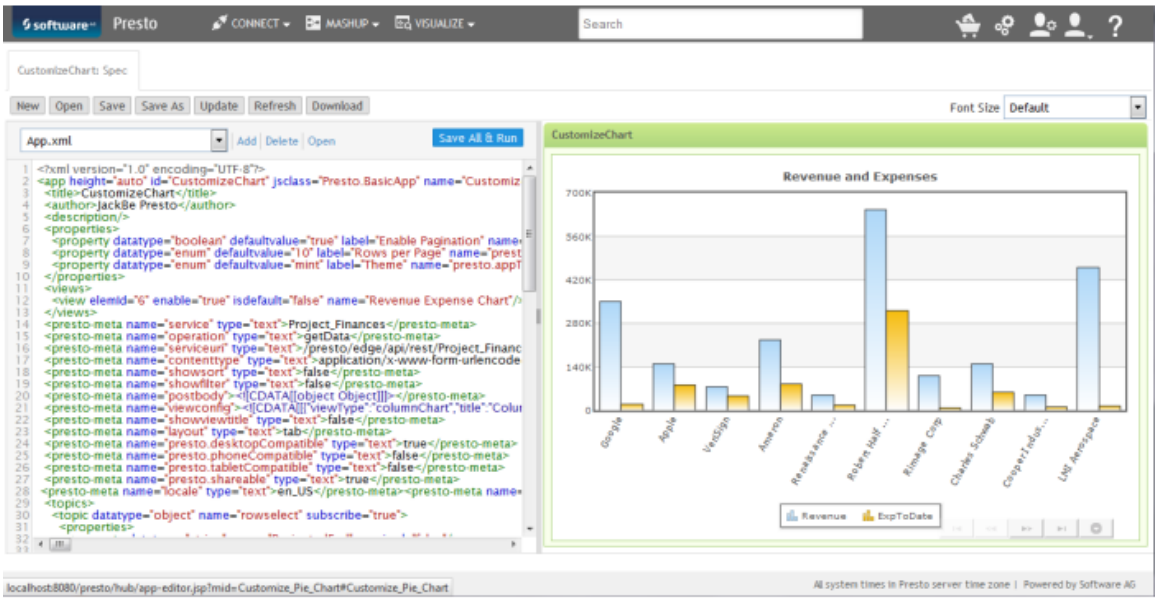
1. If you wish to customize one or more views, create a basic app with those views. See "Create a Basic App" on page 1093 for instructions.

2. Open the basic app with the view(s) you want to customize. See "Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub" on page 84 for instructions.
3. Click **Edit >** **Source** in the basic app's artifact page to open the App Editor with this basic app.

   This opens the app specification for this basic app which defines properties, views and other information needed to run this app.



   The app spec is the only resource for the basic app. For more information on app specifications, see the "App Specification Reference" on page 1298.
4. Click **Save and run** to test the app.



5. To customize the app and its view, you edit the app specification or add other resources. You can:
   - Customize Built-In Chart Attributes
   - Customize Themes for Basic Apps and Views
   - Customize the Layout for Basic Apps
6. Save and run the app frequently to verify your changes.

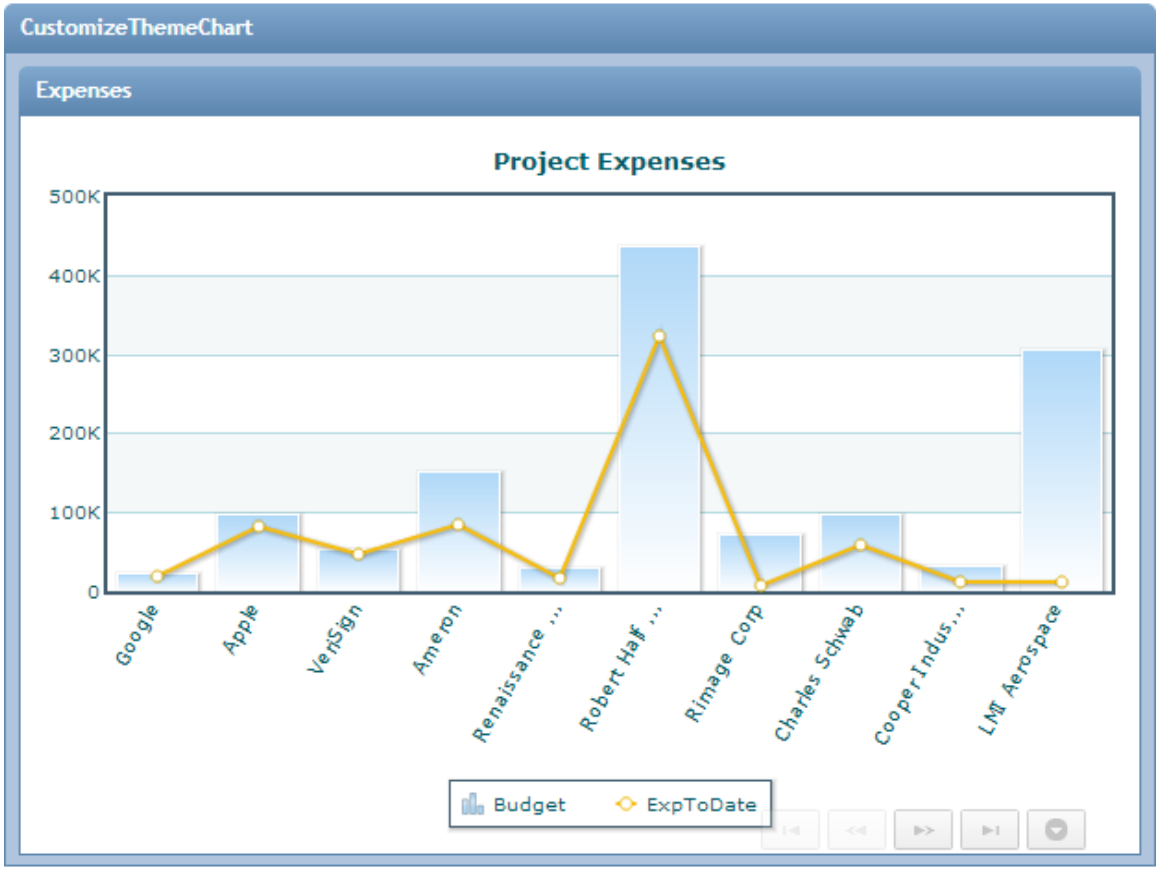
Customize Themes for Basic Apps and Views

Themes control the visual properties of:

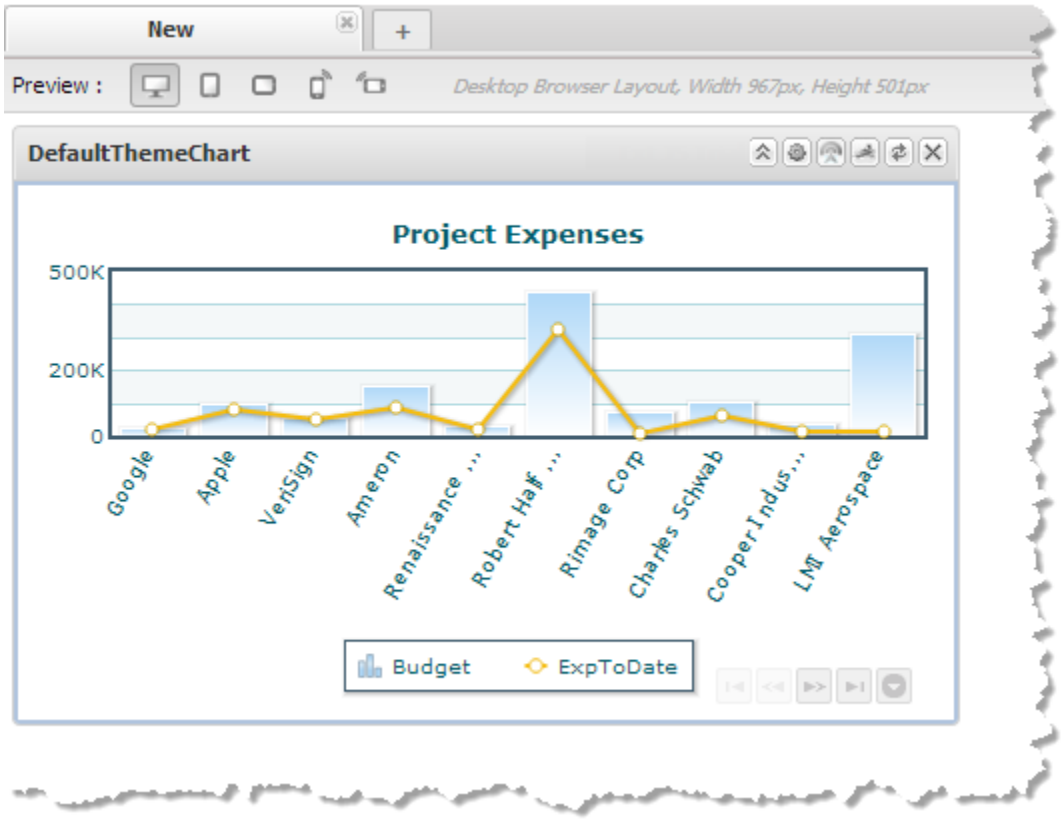
- Title bars for both the app and each view within a basic app.
- Borders and backgrounds for the App Framework that is wrapped around the views for a basic app.
- Dialogs for tools, such as forms for input parameters, and for messages.
- Icons used for toolbar buttons.

Users choose a theme from ten different colors when they create basic apps. Gray is the default theme. Users also determine whether view title bars are visible.

Themes are fully visible when basic apps are published individually such as this example:



When basic apps are included in workspace apps, however, the workspace hides both the app title bar and all view title bars by default. The workspace supplies a standard app title bar for all apps within the workspace. Users can choose to show the view and basic app title bar, but generally the impact of themes within a workspace is much more limited. This example shows the same basic app within a workspace with only a small background of the blue theme showing:

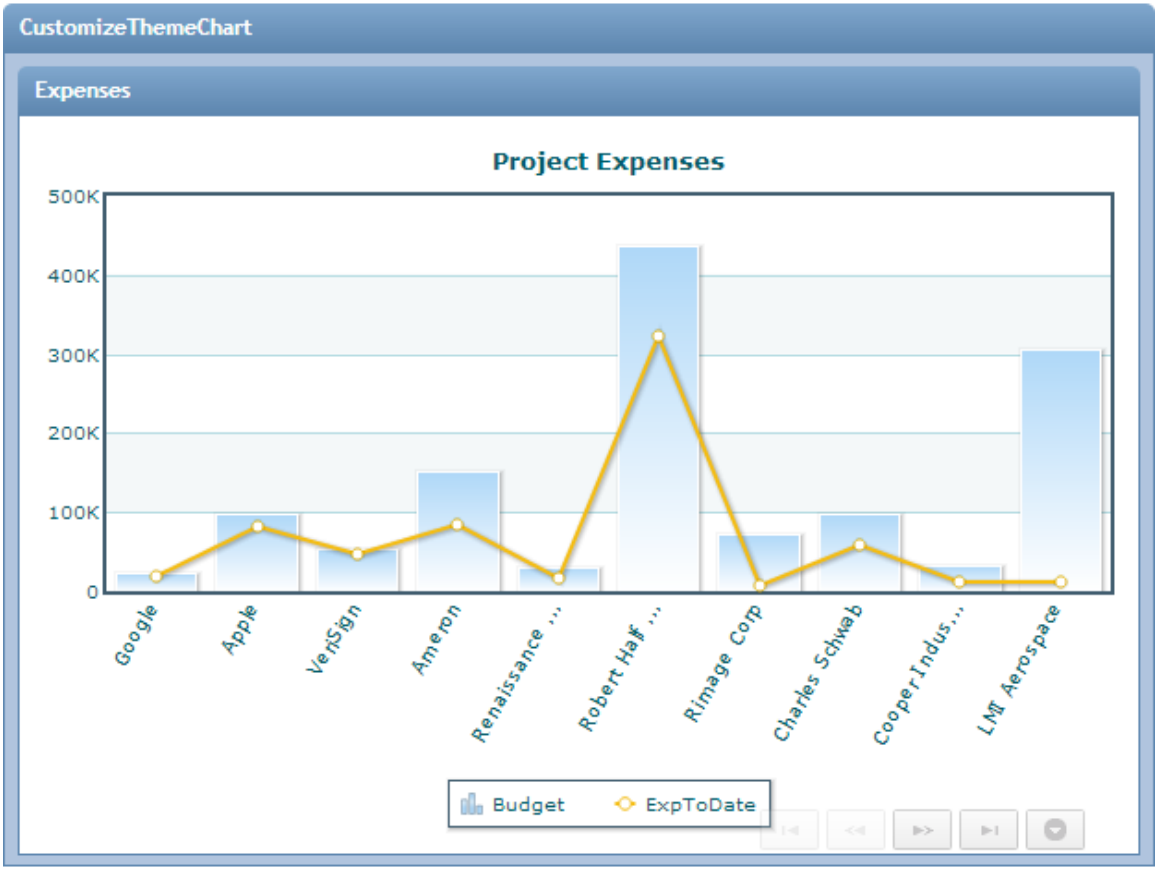


You can customize any visual property defined in the CSS for the theme used for a basic app.

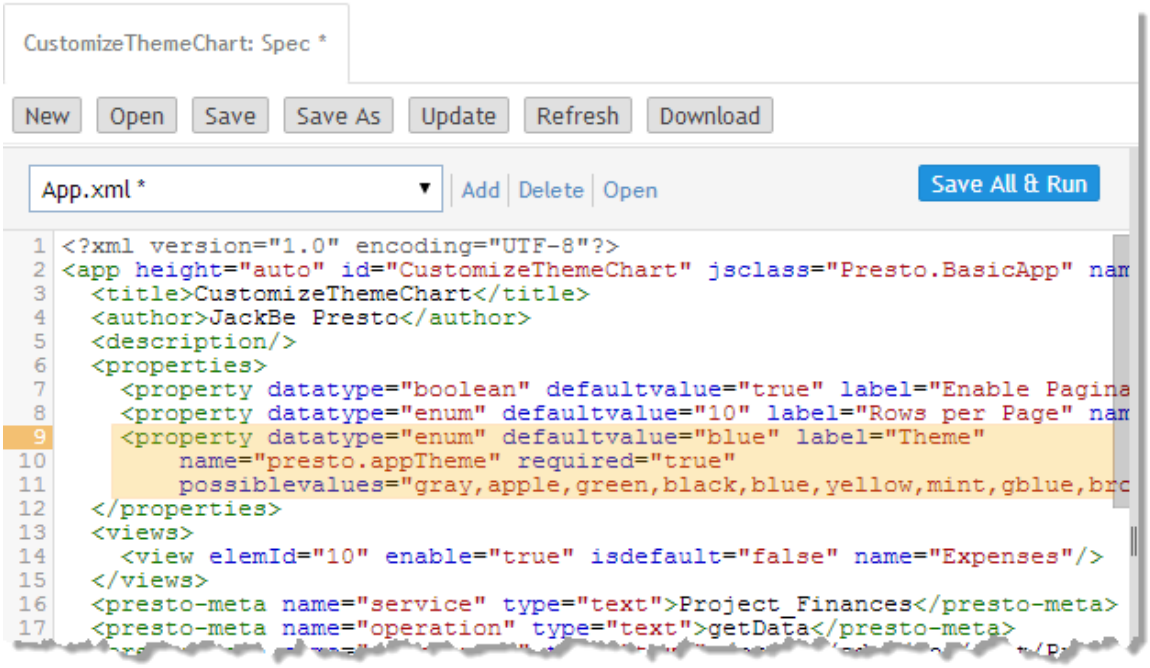
To customize a basic app's theme

1. If needed, create a basic app with the views you want to use. Open the basic app in the App Editor. See "Customize a Basic App or View" on page 1198 for instructions.

   This example uses the blue theme and a single chart view shown previously:



The App Specification for this basic app contains a `<property>` element named `Theme` which defines the built-in theme for the app as the default value:



2. Create a CSS file with the style overrides for the theme for this app.

   The CSS selectors you use make use of standard classes provided in the App Framework elements that wrap basics apps. See "App Framework for Basic Apps" on page 1292 for an illustration of these elements and class names. There are unique class names to select the app as a whole and to select each view which you can use for customization:

   - **App-specific class** = `app-`*`app-name`*, such as `app-CustomizeThemeApp` for this example. This selects the root node for the basic app including all elements provided by the App Framework.

     The app name is defined in the app specification on the `<app>` element. If the name includes spaces, spaces are transformed to underscores (_) in the class name.

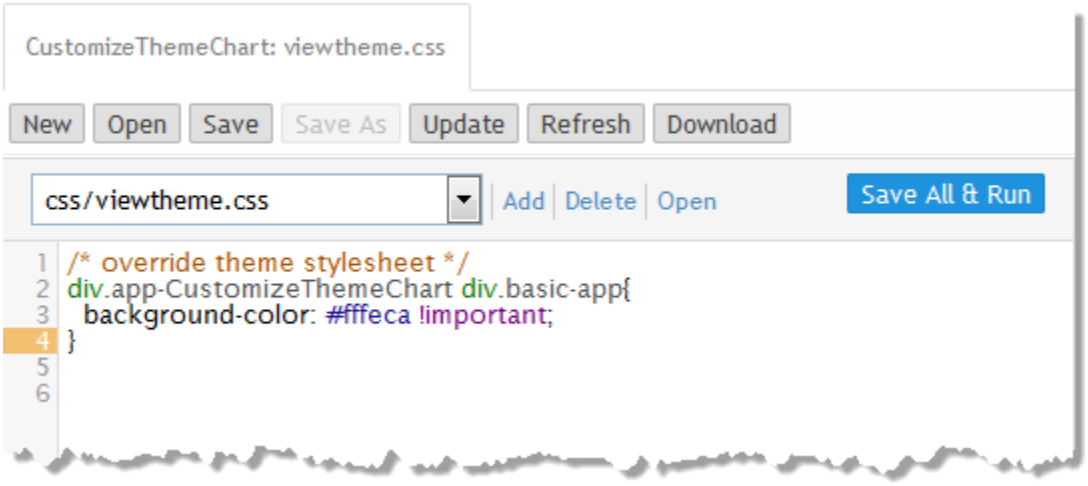
   - **All views** = `views-container` wraps all the views in the basic app.

   - **View-specific class** = `view-`*`view-name`*, such as `view-Expenses` for this example.

     View names are also specified in the app spec in `<view>` elements. Spaces in view names are replaced with underscores (_) in the corresponding class names.

   Note: There are many more selectors used in themes and other attributes you can use in selectors that are specific to basic apps. See "CSS Selectors for Themes" on page 1206 for more information.

   In this example, we are simply going to change the color of the background for the app's theme:



This uses the app-specific class `app-CustomizeThemeChart` to select the basic app and selects the element containing the content area with the `basic-app` class to apply a different background color. Use of the `!important` directive helps to ensure that this overrides the default property for the theme.

3. Add this CSS file to the basic app:

   a. Click **Add** and select **CSS** as the resource type to add.

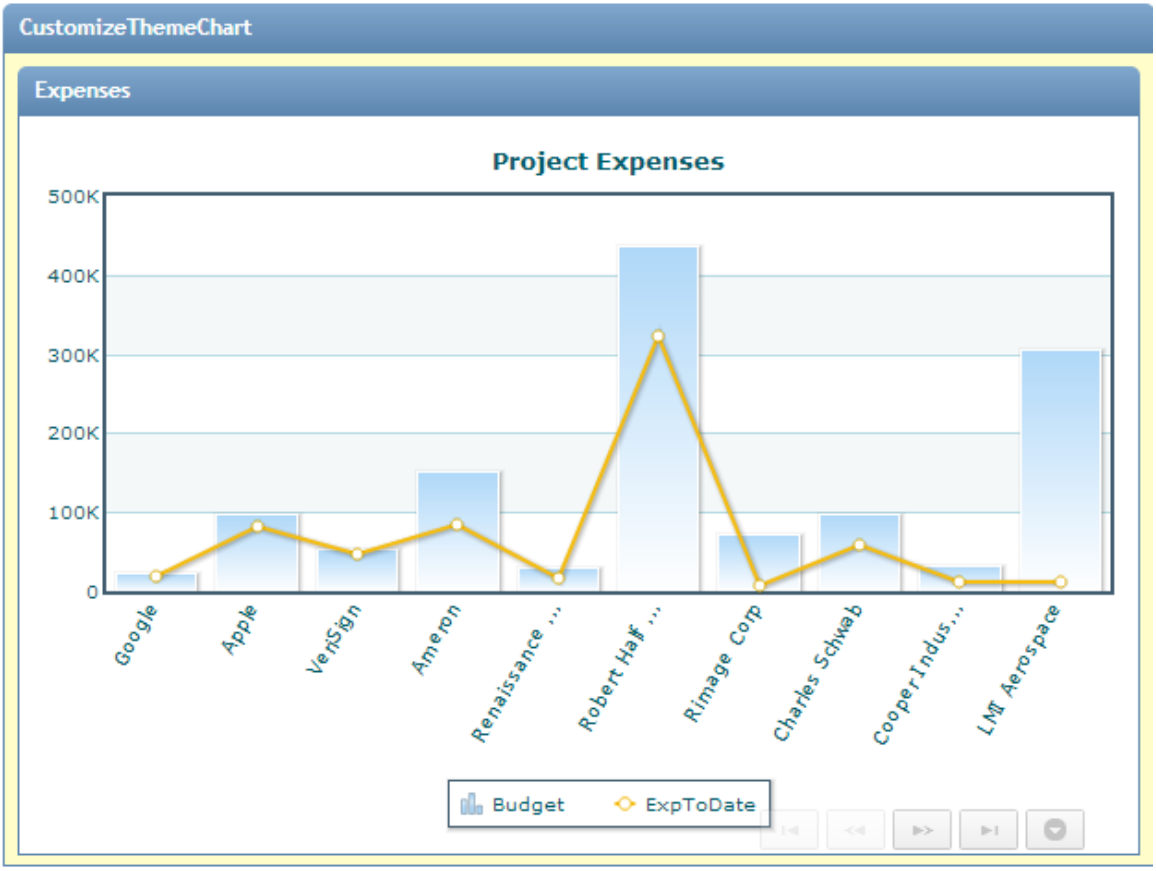
   b. Click **Browse** and find the CSS file to use to customize the theme for this basic app.

   c. Click **Add Local Resource** to upload this file and add it to this basic app.

   The App Spec for this basic app now has a new `<require>` element with the relative URL to this CSS file.

4. Click **Save and run** to apply this CSS.

   The default blue background for the theme now reflects the customized CSS:



5. Select the CSS file and update the styles as needed to further customize the theme.

   You could, for example, change the background color of the view title bar by adding a style to the CSS like this:

```
 /* overrider theme styleshee */
 div.app-CustomizeAppTheme{
  background-color: #ffeaca !important;
 }
div.app-CustomizeAppTheme div.view-Expenses .view-header{
background-image: none !important;
background-color: #ffbd58 !important;
}
```

CSS Selectors for Themes

The styles for the built-in MashZone NextGen themes for basic apps are defined in the *MashZoneNG-install* /apache-tomee-jaxrs/webapps/presto/hub/css/app-themes.css file.

Some general rules of thumb for finding and working with theme selectors:

- Style names with selectors starting with `div.mashlet.`*`color`*`-theme` apply to the theme for that specific color.

- Style names with selectors that do not have a color name are for the Gray theme, which is the default theme.
- Use `div.app-app-name` in override styles for either `div.mashlet.color-theme` or `div.mashlet`.
- Icons for toolbar buttons are defined as 16 x 16 pixels.

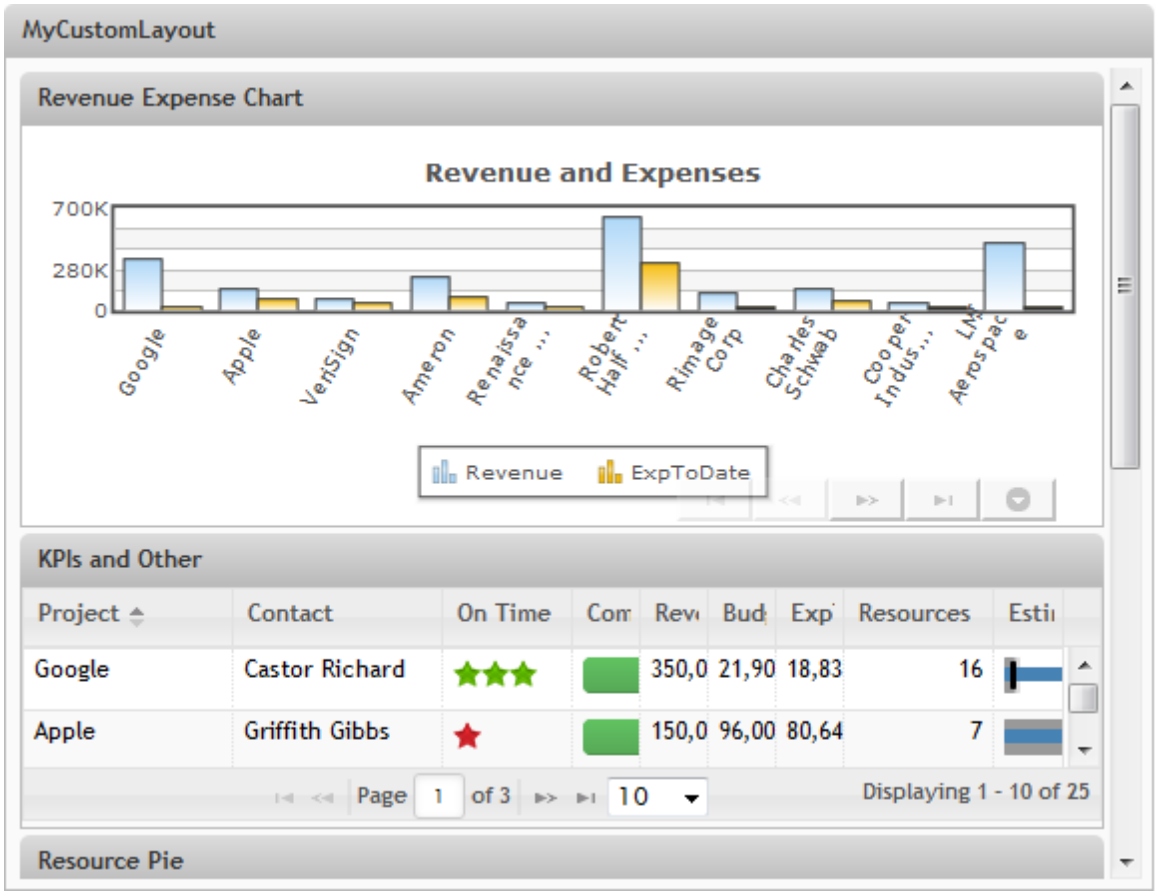
  Images are defined in styles with `.app-chrome-toolbar` for the app title bar or `.view-header` for view title bars.
- Title bars are defined in styles with `.views-container .view-header`, for views, or `.data-view .view-header`, for the app.

  You can override title bars for all views using a style with `.views-container .view-header`. Or use `.view-view-name .view-header` to override the view title bar for a specific view.
- Borders are defined in styles with `.mashlet-content`, `.view-card` and `.basic-app-input`.
- The content background is defined in styles with `.mashlet-content`.
- Tabs for multi-view apps with a tab layout are defined in styles with `ul.ui-tabs-nav`.

Customize the Layout for Basic Apps

Basic apps that include two or more views have a layout that determines where each view is rendered within the app. Views can render in separate tabs or cards that users navigate to or they can render all views within the single card of the app in simple geometries such as side-by-side.

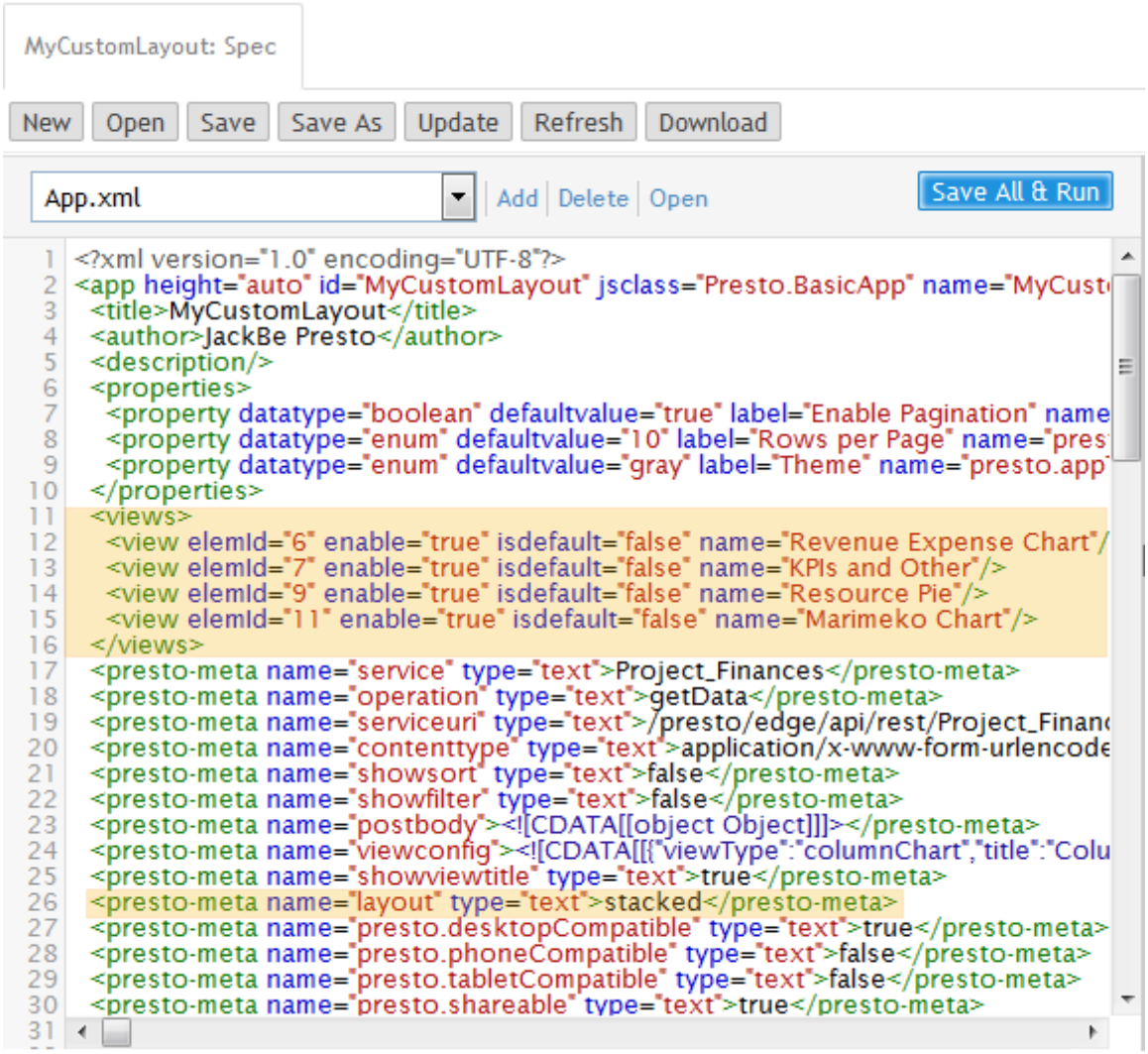
You can provide a custom layout for basic apps for more complex layouts. The example shown in this topic is a basic app with four views that are rendered in a single stack:



To customize this layout

1. If needed, create a basic app with the views you want to provide in a custom layout and open this basic app. See "Customize a Basic App or View" on page 1198 for instructions.

   The App Spec for the basic app contains metadata for the views contained in the app and also the built-in layout you selected when you created the app.



2. Create an HTML fragment in a text editor of your choice with `<div>` container elements to wrap all the views and contain each view in position within the custom layout. This fragment must:

   - Have a root `<div>` container. It is a good practice to assign a unique ID or your own unique CSS class. For example:

     ```
     <div id="myCustomAppLayout"></div>
     ```

   - Have a `<div>` container for all views with a `data-presto-role="output"` attribute. For example:

     ```
     <div id="myCustomAppLayout">
       <div data-presto-role="output"></div>
     </div>
     ```

     Data roles identify the different purposes for containers within the layout. In this case, `output` indicates that the contents are meant to be rendered. See "Layout

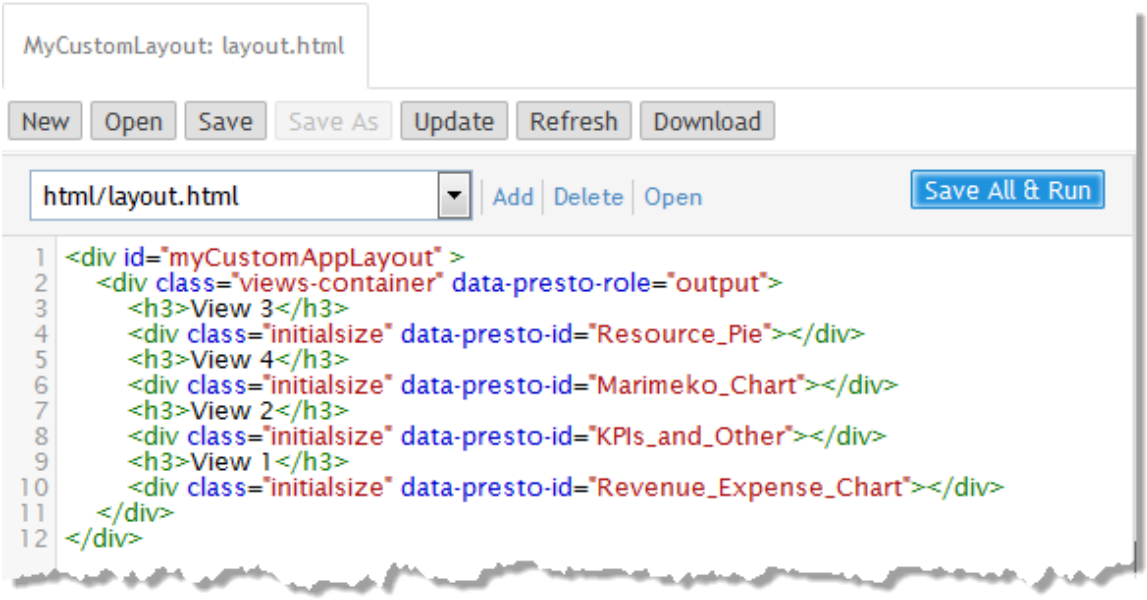
Roles" on page 1217 for a list of valid data roles for MashZone NextGen basic apps.

- Have a `<div>` container for each view with a `data-presto-id` attribute identifying the view to be rendered in that container.

  You can determine the view IDs from the names of the `<view>` elements in the App Spec. Replace any spaces in the view names with underscores (_).

  For this example, we will rearrange the views so that the Resource Pie and Marimeko Chart views appear side-by-side at the top of the app followed by the KPIs and Others view and then the Revenue Expense Chart view.

  The initial HTML for this layout looks something like this:



- Save this fragment to your computer. In this example, it is layout.html but you can use any file name.
- If your basic app has input parameters or needs to support sorting or filtering, see also "Including Built-in Tools in a Custom Layout" on page 1214 for additional steps.

3. Add the HTML custom layout to your basic app:
   a. In the App Editor, click **Add** and select **HTML** as the type of resource.
   b. Click **Browse** and find the custom HTML layout file you created previously.
   c. Click **Add Local Resource**.

      The layout.html file is uploaded to MashZone NextGen and associated with this basic app. It opens in the App Editor.
4. Update the App Spec to use this custom layout:

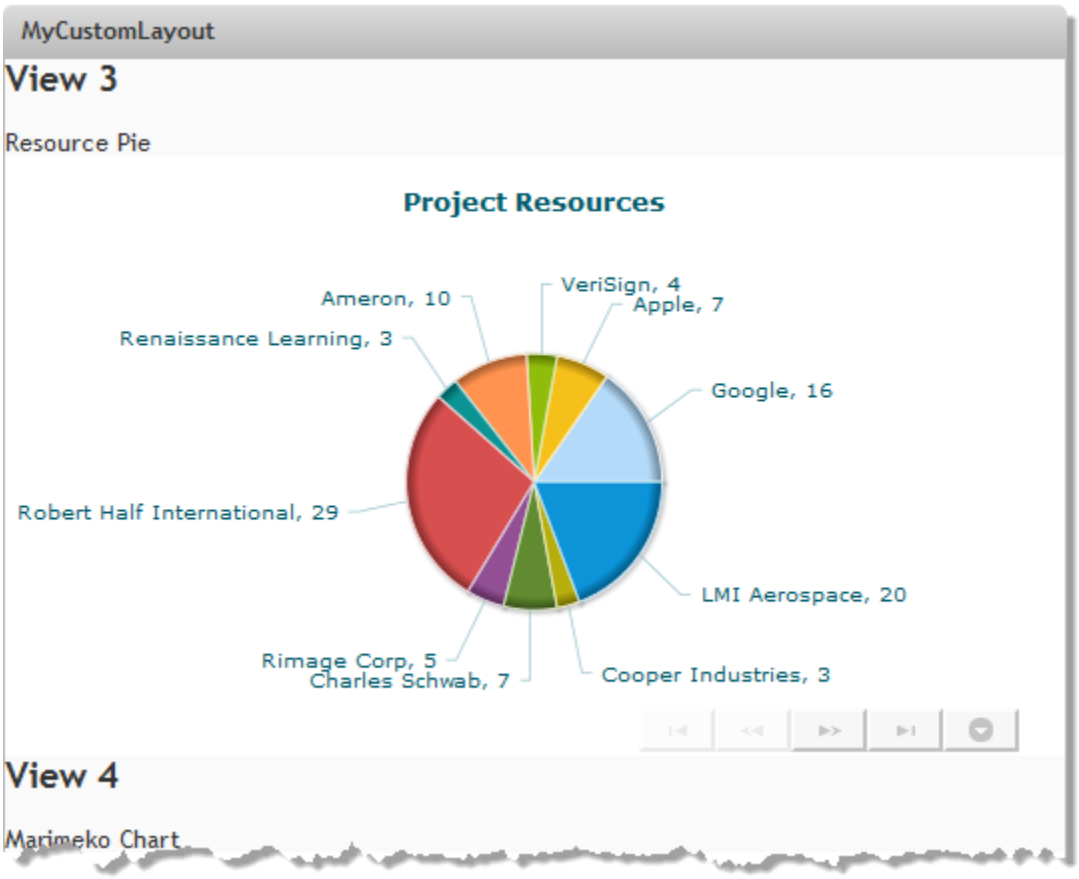
a. Use the pull-down list for files in the App Editor to move back to the App Spec.

b. Change the value of the `<presto-meta name="layout">` element from `stacked` (or whatever built-in layout is assigned) to `custom`.

   Once you change the layout metadata for the basic app to `custom`, MashZone NextGen uses the HTML file associated with the basic app to control the content and layout for the app.

5. Click **Save all and run** to save this change and test the app.

   This example now looks like this:

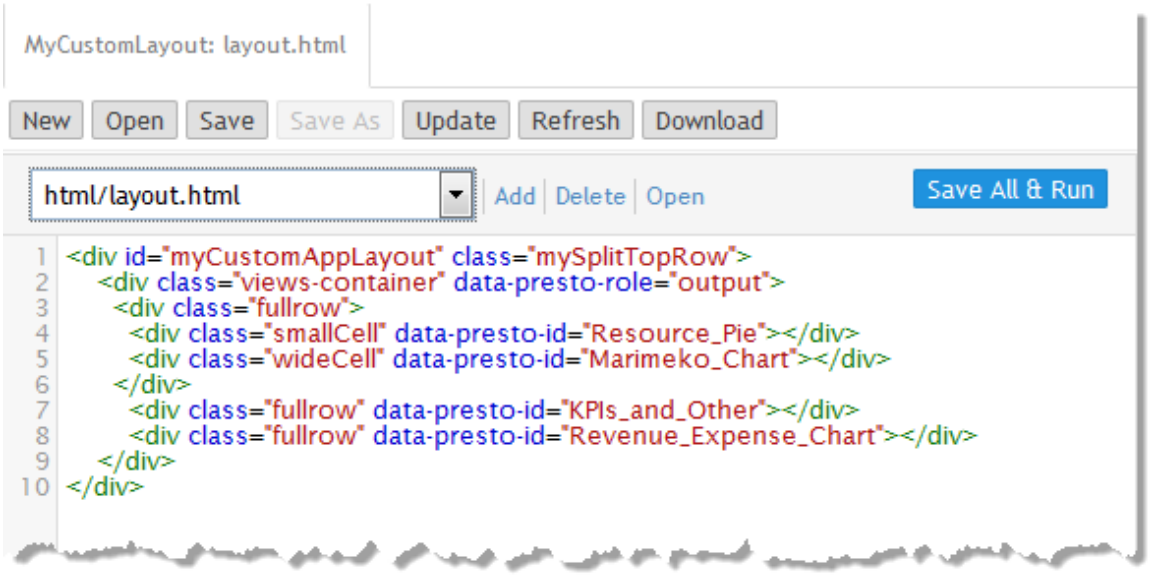


The app title bar looks just the same. But the views have been rearranged in the order set by the custom layout. The headings from the custom layout also appear, such as `View 3`. So do the view titles, such as `Resource Pie`, that come from the basic app, but the default look and feel has disappeared for view headers.

This is because custom layouts override the default containers and classes for the contents of the basic app. The containers and classes for the App Framework that wrap the basic app, however, are still present allowing the app title bar to remain untouched. For an overview of the standard container elements and classes for a basic app, see "App Framework for Basic Apps" on page 1292.

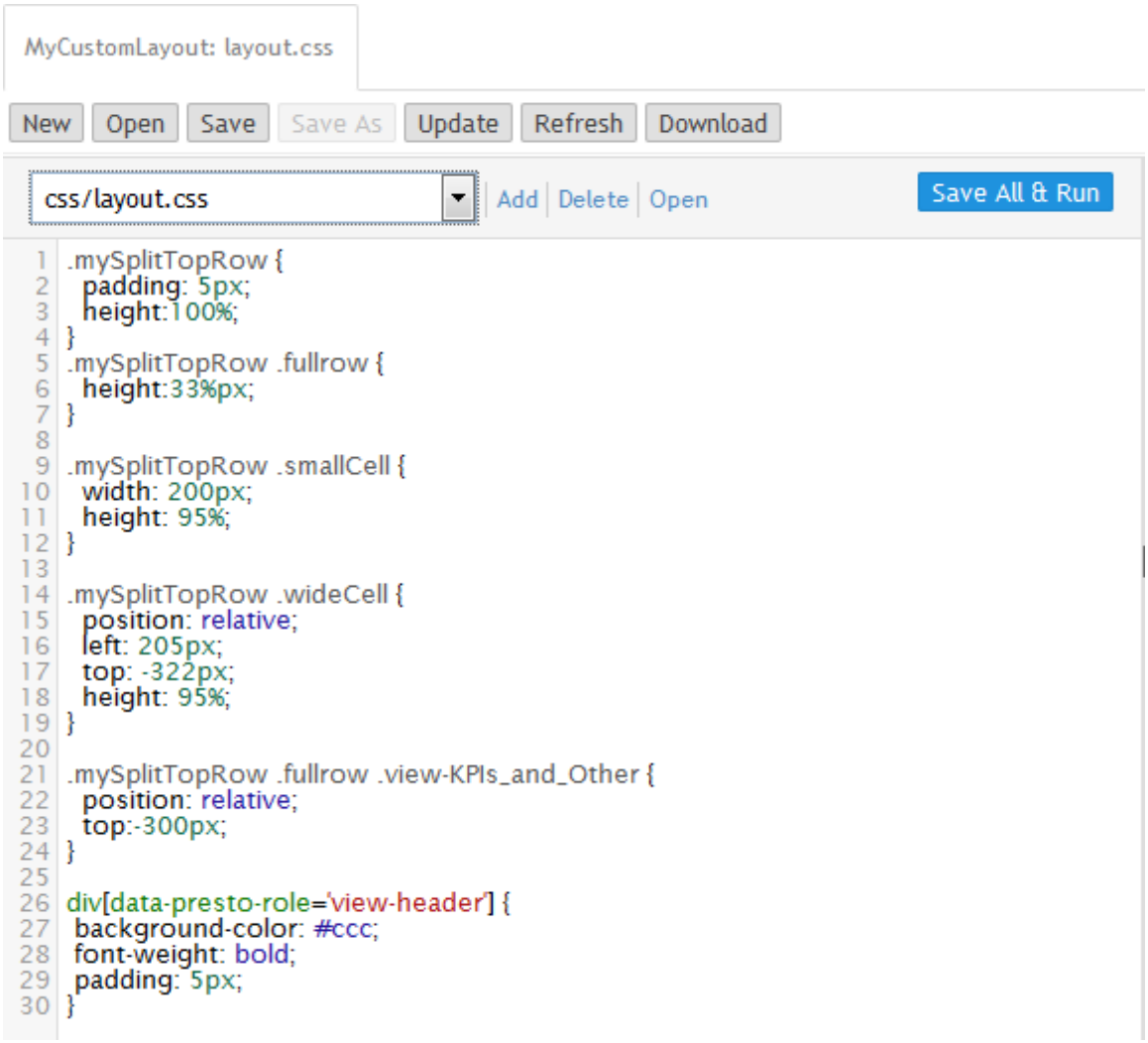
6. Add CSS classes to better control the layout.

   It is a best practice to assign unique classes to the various containers in your custom layout and add CSS styles to control the look and feel. For this example, we will simplify the layout a little and add classes to make layout.html be:



7. Create a CSS file with styles to control the layout and look and feel for your custom layout and save this file.

   For this example, the file is named layout.css and the CSS rules are:



Most of the styles use the class names assigned to containers in layout.html. However, the style rule that is applied to the view title bars uses a selector based on `[data-presto-role='view-header']`. This data role is used for the view headers generated by the App Framework.

8. Upload the layout CSS file:

   a. Click **Add** and select **CSS** as the type of resource.

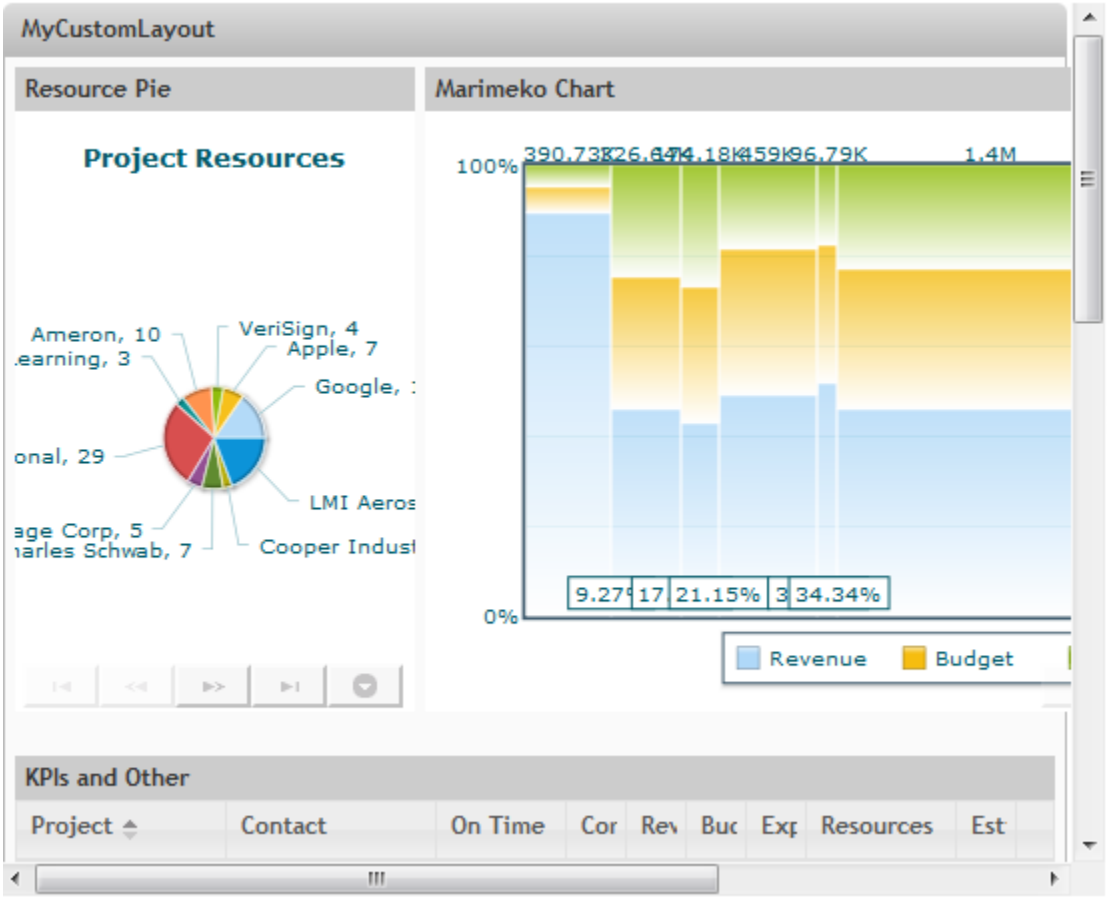
   b. Click **Browse** and find the custom CSS layout file you created previously.

   c. Click **Add Local Resource**.

      The layout.css file is uploaded to MashZone NextGen and associated with this basic app. It opens in the App Editor.

9. Click **Save all and run** to save this change and test the app.

   The basic app now renders the Pie and Marimeko views in a singe line, followed by the other views:



The view title bars also have better visibility and match more closely with the app title bar.

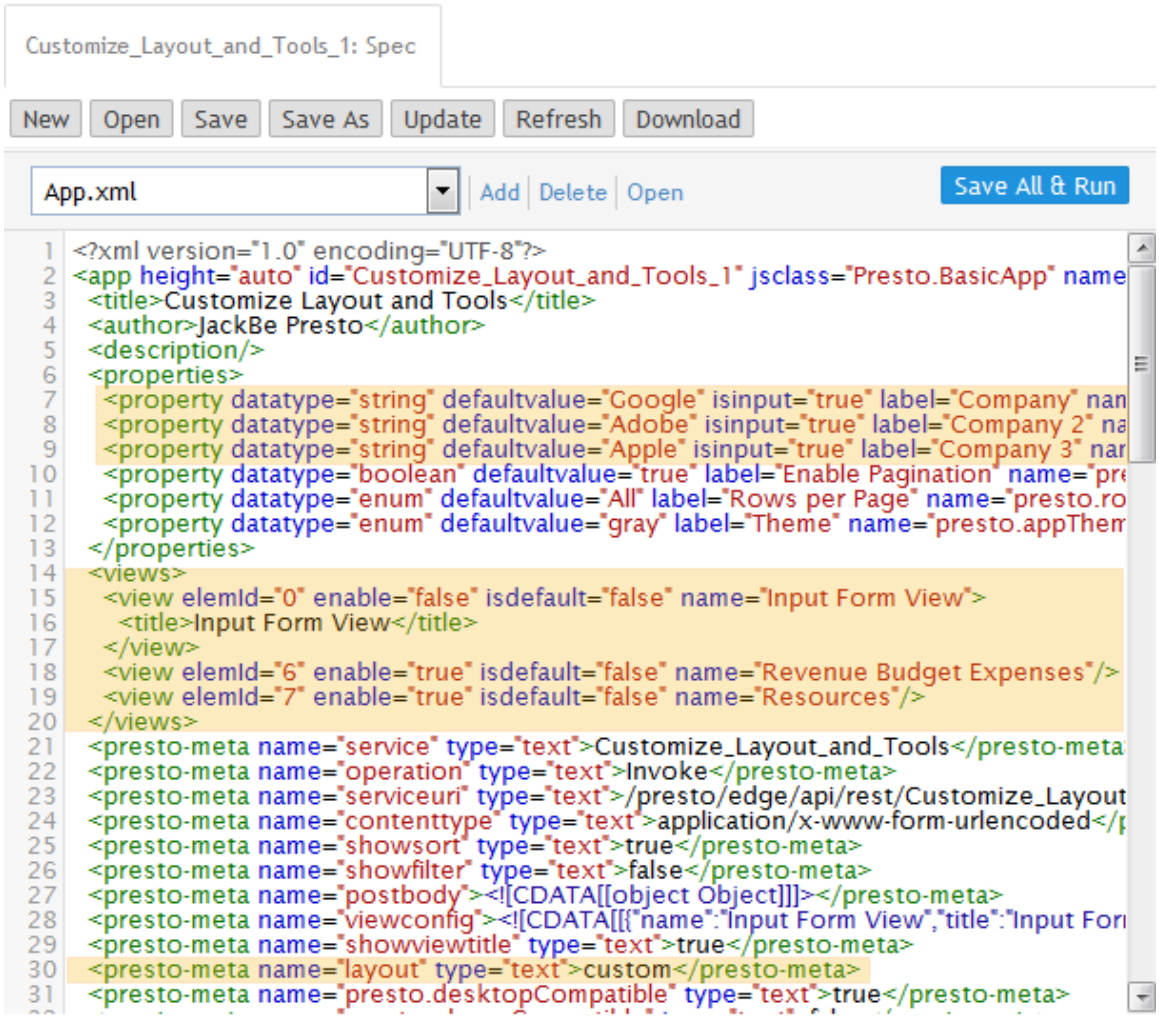
10. Update the HTML and CSS as needed until you are satisfied with the app.

Including Built-in Tools in a Custom Layout

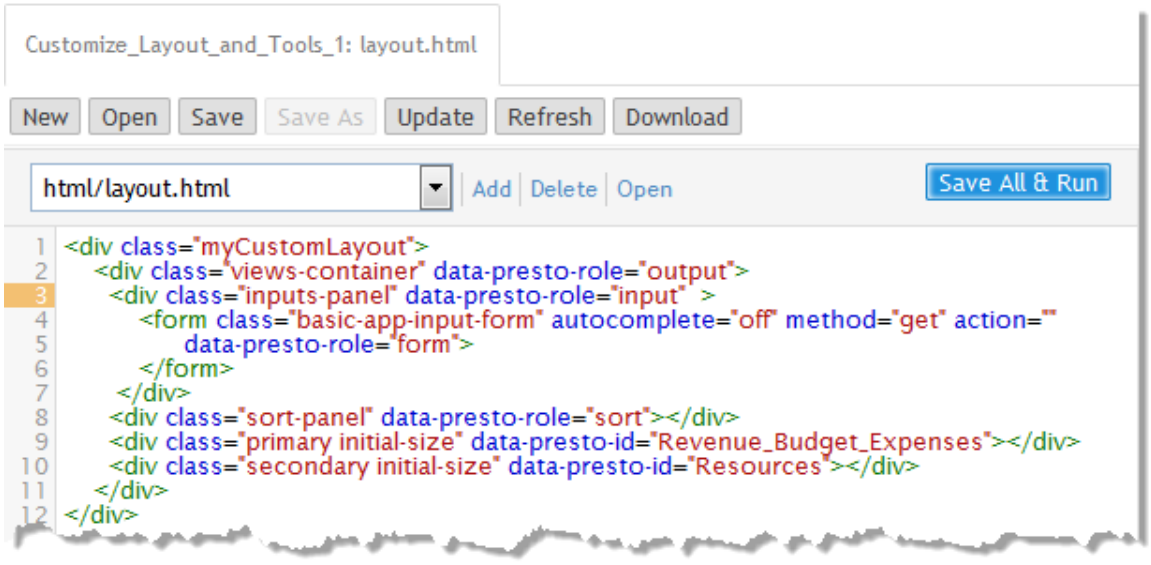
If the basic app that you are customizing has input parameters or you have enabled sorting or filtering tools for the app, you must add support for these forms in the HTML for your custom layout. You can also add support for the standard message overlay that typically displays messages while app data is loading.

To add support for tools, you add `<div>` or `<form>` containers to the layout HTML with the appropriate value for the `data-presto-role` attribute.

The app for this example has input parameters and has sorting enabled. The App Spec contains `<property>` element for each input parameter. It also has a `<view>` element for each view and one for the input parameter form. This input form view is disabled, causing the input form to render as a dialog.

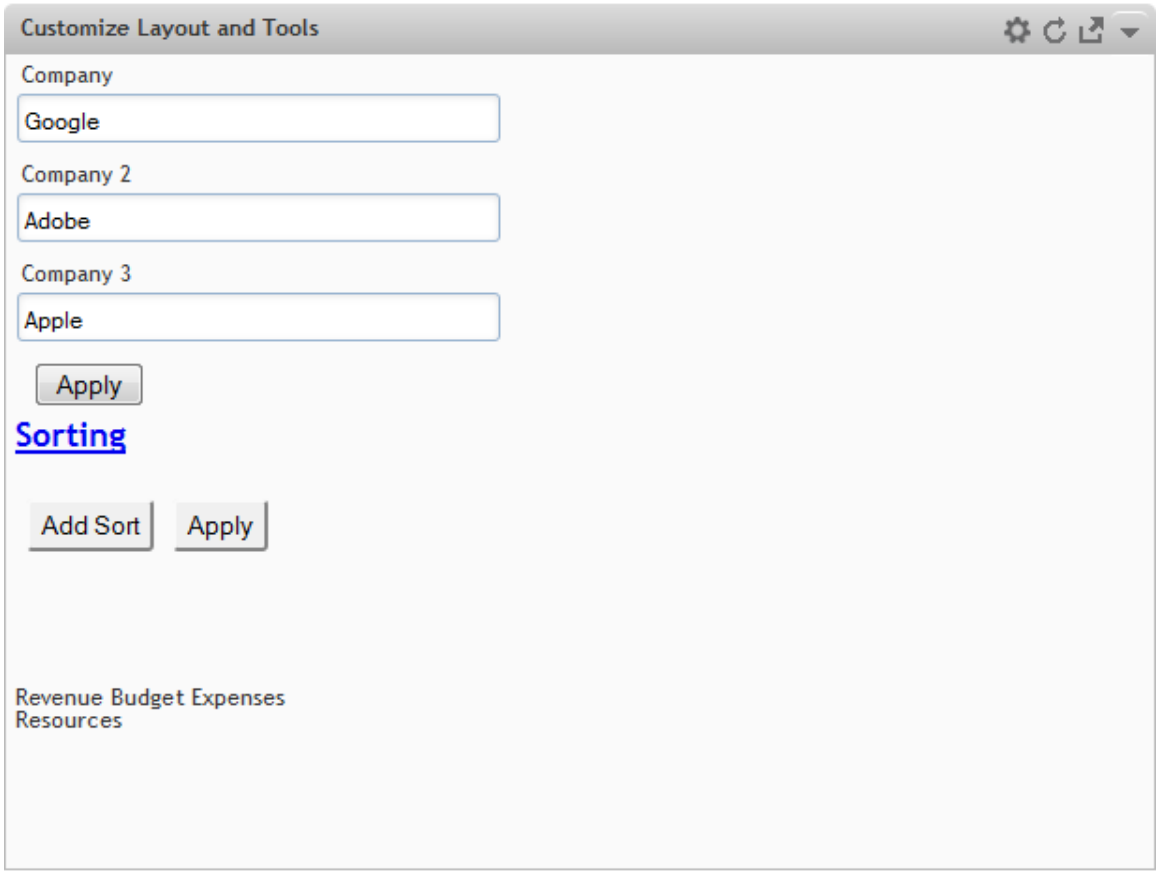


The custom layout for this app is shown below:



In addition to the `<div>` containers for each view to render, this layout includes a `<form>` for the input parameters forms with a data-presto-role of `form`. It also includes a `<div>` for the sorting tool form with a data-presto-role of `sort`.

This forms render when you run the app:



Supply input parameters and click **Apply** to run the mashable or mashup for the app and allow the views to display the results.

Layout Roles

dataview	Automatically generated container for each view. Contains view-header and view-card.
form	Optional `<form>` to generate the standard input parameter form for this app. Can be contained in and input container or
input	Optional container to wrap the `<form>` for input parameters for this app.
filter	Optional container to generate the standard filter tool form for this app.

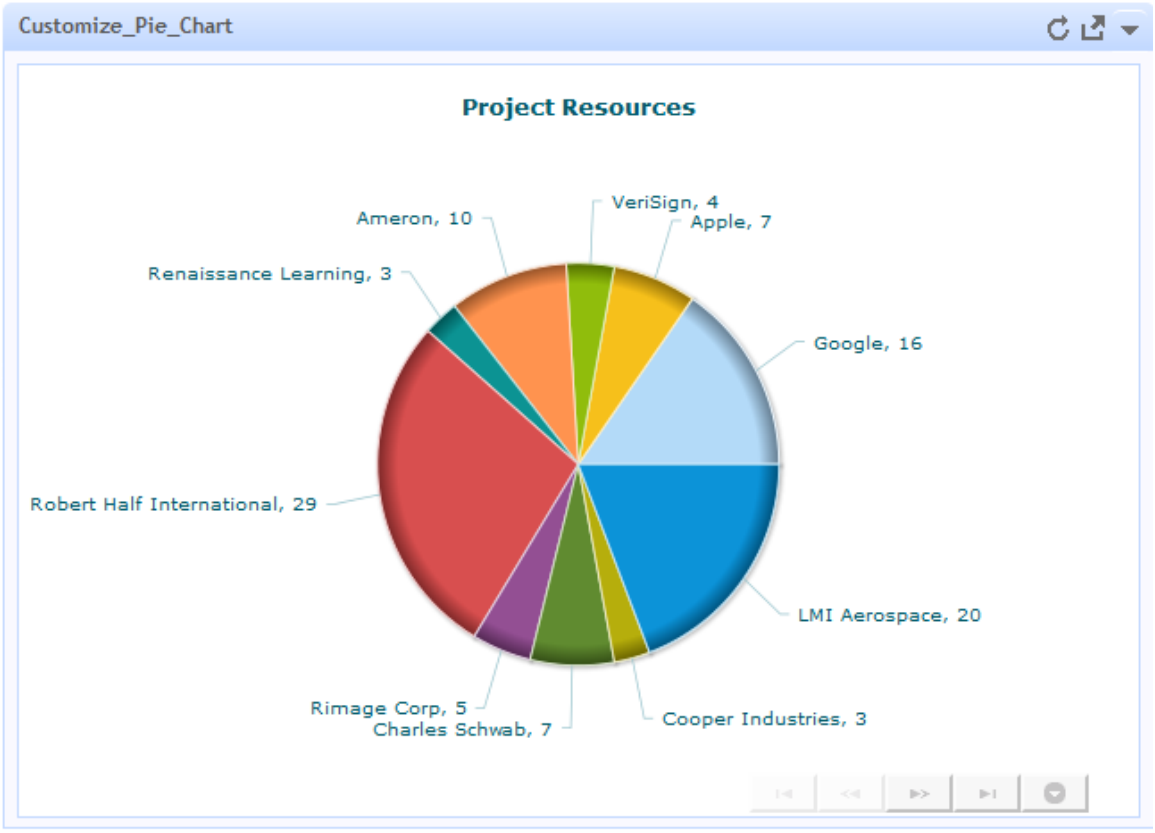
output	Required role. This container must contain s for each view to be rendered. Optionally can also contain s for tools for the app (input, filter, sort or overlay).
overlay	Optional container for a messages container that will overlay app content for messages.
sort	Optional container to generate the standard sort tool form for this app.
title	Automatically generated container for the text of the view title.
view-card	Automatically generated container for the body of the view. Contains view-container.
view-container	Automatically generated for each view. This container is where the view will be rendered.
view-header	Automatically generated container for the title bar of each view. Contains title.

Customize Built-In Chart Attributes

MashZone NextGen charts and gauges use FusionChart libraries to create the built-in charts, with the exception of real-time charts. FusionCharts libraries use a set of attributes to define the configuration options for a chart.

The MashZone NextGen View Wizard adds FusionChart attributes to view configuration when you create views, along with additional information used by MashZone NextGen. You can add these FusionChart attributes, remove these attributes or edit these attributes in view configuration information to customize the look of chart and gauge views.

To make these attributes easy to work with, you must move this configuration from the app spec for a basic app containing this view to a separate JSON file. This topic presents an example of that process using a simple pie chart:



By default, the pie chart uses *smart labels* which are call outs (simple lines and labels). This works very well when there are quite a few pie slices, but may look very bare with fewer slices. We're going to change this pie chart to use a legend rather than smart labels.

The MashZone NextGen View Wizard doesn't allow you to configure a legend for pie charts. To use a legend in this case, you must customize the attributes that the MashZone NextGen View Wizard generates for the view:

1. Create a basic app for the view you want to customize and open it in the App Editor. See "Customize a Basic App or View" on page 1198 for instructions.
2. "Move the View Configuration to a JSON File" on page 1219. The view configuration contains the attribute settings for Fusion Charts for this view. Moving this configuration to a separate file makes it easier to edit these attributes.
3. "Customize the Fusion Chart Attributes" on page 1221 as needed. For this example, we will add a legend and remove the smart labels for this view.

Move the View Configuration to a JSON File

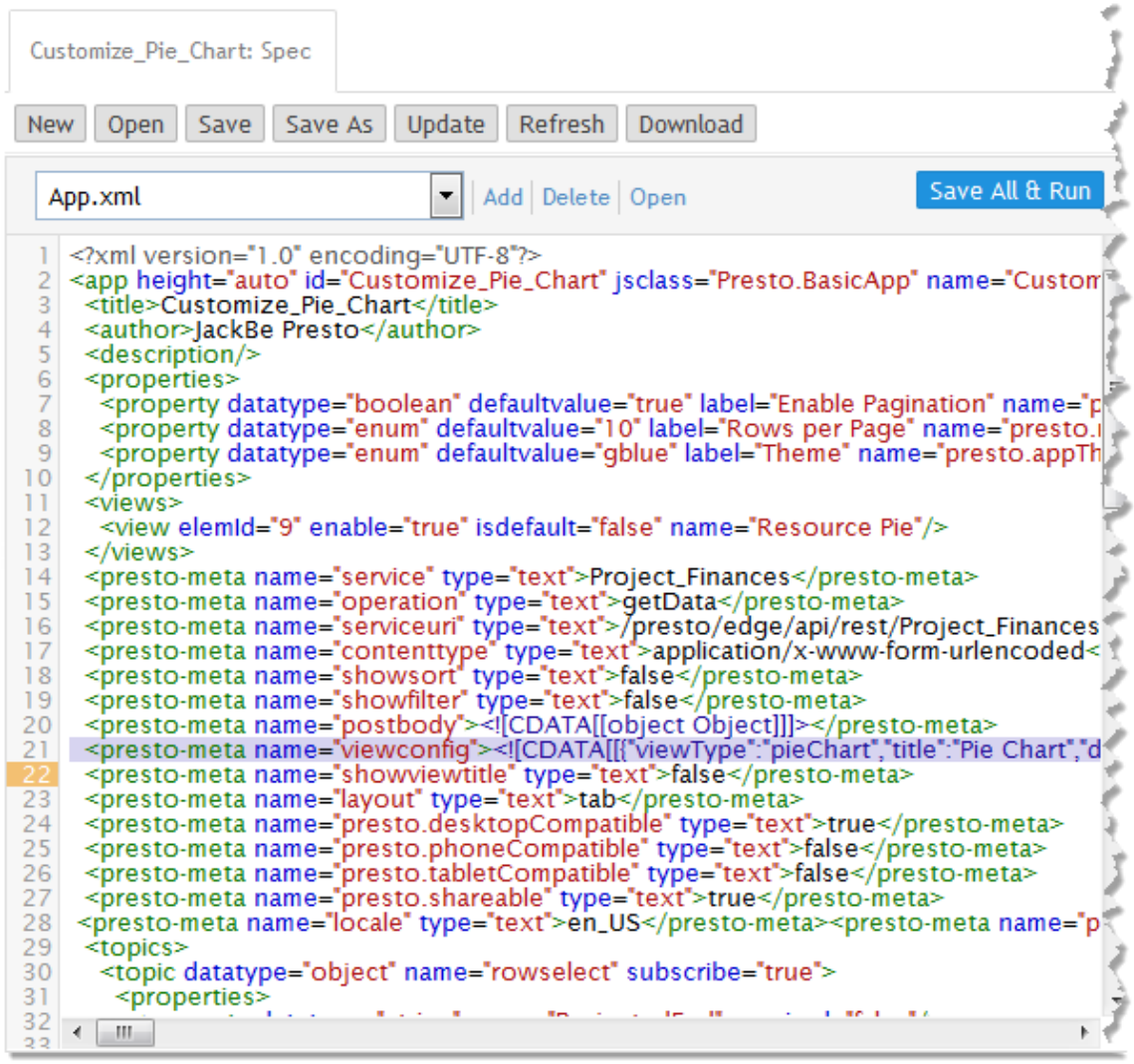
View configuration information is defined in a `<presto-meta>` element named `viewconfig`. The contents of this meta-data element is a JSON object, with the configuration information for one view. Because apps can have multiple views, the JSON configuration object for each view is one member of an array. This array is

wrapped in a CDATA marked section to ensure that the JSON syntax does not cause parser errors for the app spec.

Disected, this would look like this:

```
<presto-meta name="viewconfig">
  <![CDATA[
    [
     {...JSON config object for one view...},
     {...another view config object...}
    ]
  ]]>
</presto-meta>
```

In the App Editor, you see one line for this configuration:



To move this configuration to a separate file:

1. Copy the line from the app spec with `<presto-meta>` for the view configuration.

2. Paste this in a blank file in a text editor of your choice.
3. Delete the `<presto-meta>` start and end tags.
4. Delete the start and end tags for CDATA (`<![CDATA[` and `]]>` being careful to *retain* the inner bracket symbols, [and], for the array with this JSON ogject.
5. Save this as a file named viewconfig.json in any folder on your computer.

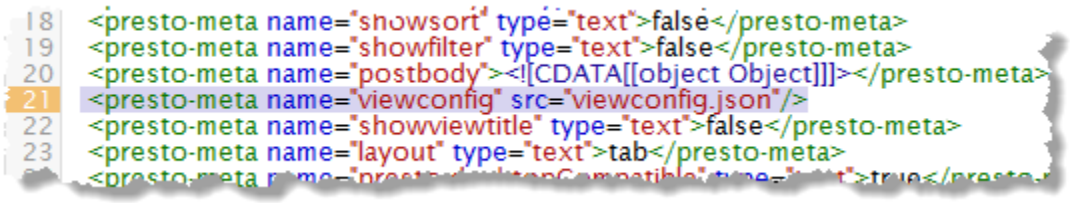
> **Note:** This file should not be treated as a JavaScript file with `.js` as the file extension as this causes errors.
>
> It is also a good practice to validate this JSON using tools such as jslint.

Once you have the JSON configuration in a file, you can add this to the basic app to make it easier to edit the chart attributes:

1. In the App Editor, click **Add**.
2. Click **Other** as the type of resource.
3. Click **Browse** and select the viewconfig.json file you created previously.
4. Click **Add Local Resource** to upload this file and add it to this basic app.
5. Edit the `<presto-meta name="viewconfig">` line in the app spec to replace the in-line configuration and refer to the viewconfig.json file:
   a. Delete the content of the `<presto-meta>` element for this line.
   b. Remove the end tag and make this an empty tag, such as:

      ```
      <presto-meta />
      ```

   c. Add a `src` attribute and set the value to viewconfig.json. This line should now look like this:



6. Click **Save and run** to save and test these changes.

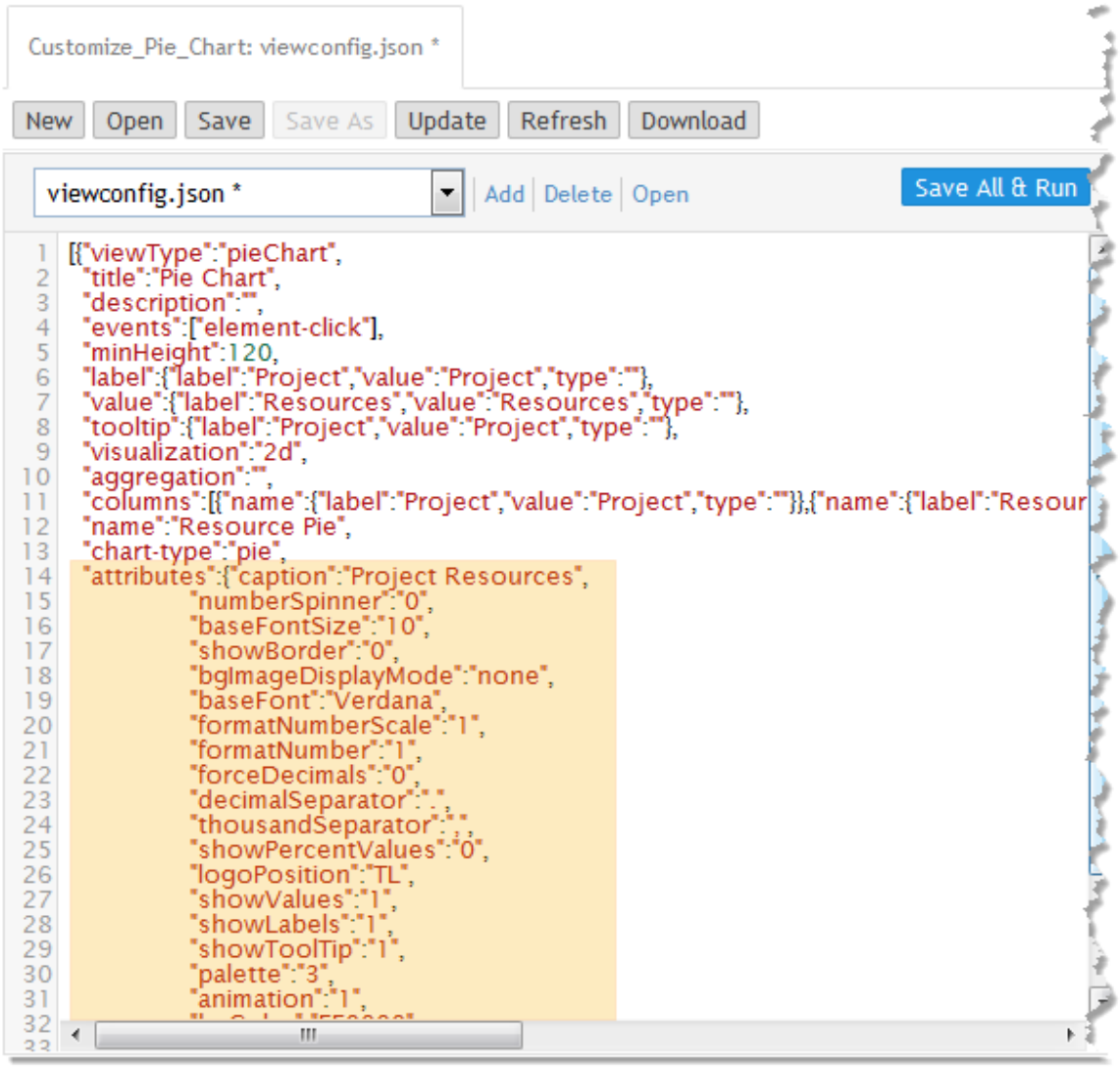
   The app should display exactly as it did originally in the preview pane.

You can now begin editing viewconfig.json to tweak chart attributes as needed.

Customize the Fusion Chart Attributes

Once you are sure the viewconfig.json file is working, choose viewconfig.json from the file drop down in App Editor and break the JSON configuration up with whitespace,

tabs and multiple lines to make this configuration easier to understand. The JSON for the pie chart example looks something like this:



The specific properties you see in view configuration varies depending on the type of chart and configuration options that users chose when they created the chart. The information in this JSON is used by MashZone NextGen and in some cases, passed directly to Fusion Chart functions.

The `attributes` property, in particular, is an object that contains Fusion Chart attributes that you can add to, remove or edit to take advantage of functionality not supported by the MashZone NextGen View Wizard. Initially, it contains both default property values and properties users chose when the view was created.

For information on Fusion Chart attributes, see documentation for:

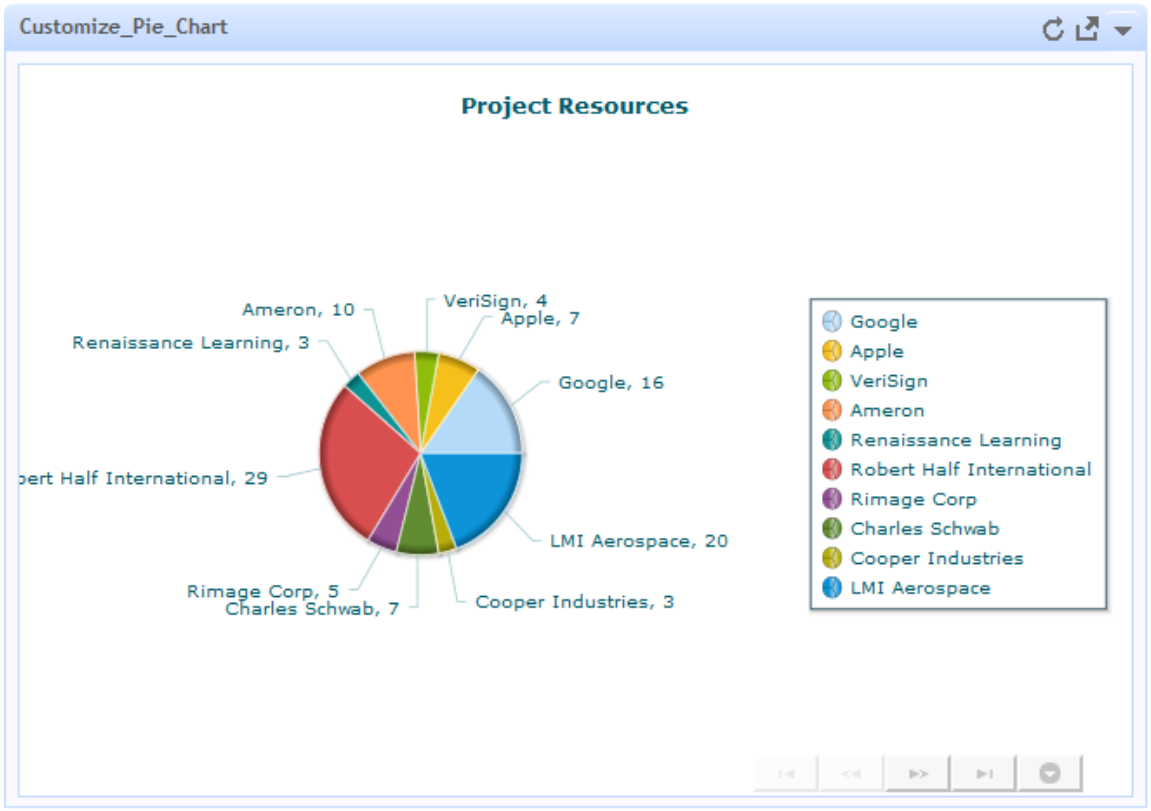
- FusionCharts XT 3.3.1

FusionWidgets XT 3.3.1

For this example, first we add properties to display a legend. In the `attributes` property, add the following lines:

```
...
  "showTooltip":"1",
"showLegend":"1",
"legendPosition":"RIGHT",
"legendMarkerCircle":"0",
"reverseLegend":"0",
  "palette":"3",
...
```

Click **Save and run** and the chart should look like this:



This adds the legend but the smart lines and labels are also still there. To remove the smart lines and labels:

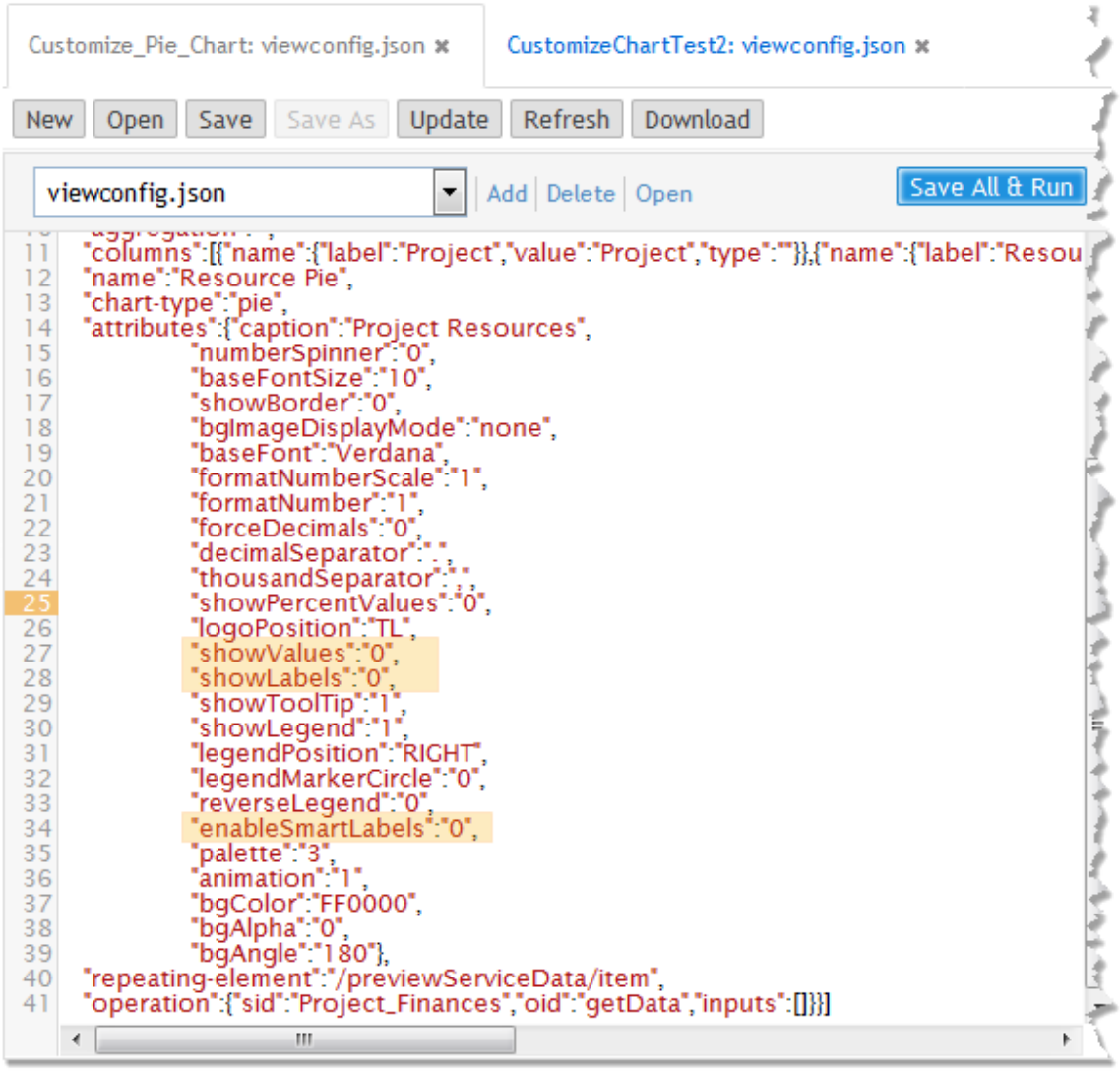
1. Add an `enableSmartLabels` attribute:

   ```
   ...
     "reverseLegend":"0",
   "enableSmartLabels":"0",
     "palette":"3",
   ...
   ```

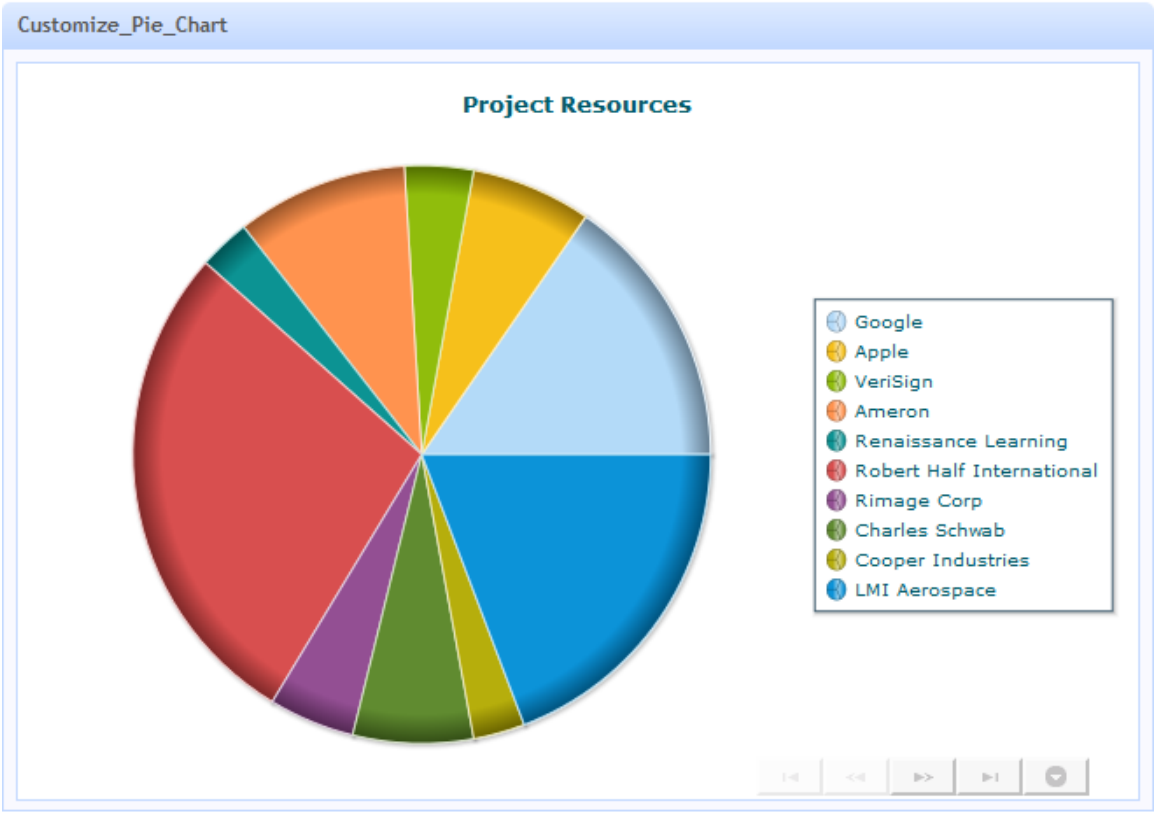
   Setting this attribute to `0` turns this feature off.

2. Edit both the `showLabels` and the `showValues` attributes and set them to `0` to turn them off.

The viewconfig.json now looks like this:



Click **Save and run** again and the chart now has a legend with no smart lines:



Create Fully Custom Apps in the App Editor

Fully custom apps can use one or several mashups or mashable information sources from MashZone NextGen. They can also directly access web services from the Internet or use widgets or gadgets that are rendered wtih <object> tags.

The requirements for the app user interface are typically what determines whether an app must be fully custom or simply can be customized based on a basic app. You may need to create a custom app to provide specific interactions in the user interface, to get specific look and feel requirements or if the app uses a form.

Custom apps can be compatible with desktop browsers, mobile phones or mobile tablets. See "Custom Mobile App Requirements" on page 1193 for more information.

This topic covers how to "Create Custom Apps from the Base App Package" on page 1226 in the App Editor. See "Working in the App Editor" on page 1194 for tips on editing and updating apps.

See these topics for basic techniques and simple custom app examples:

- "Declare, Get and Set Properties in Custom Apps" on page 1227
-

- "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502
-
- "Paginate Mashable or Mashup Responses" on page 1491
- "App Dimensions and Resizing" on page 1236
- "Enable User-Initiated or Automatic Refreshes" on page 1243
- "Wiring App Interactions" on page 1249
- "Handle Exceptions" on page 1247

For general techniques or information, see also "The Structure of a MashZone NextGen App" on page 1290, "App Packages and App Files" on page 1318, "App Specification Reference" on page 1298, "Parameters for App URLs" on page 1294and "Override Browser Caches for Updates to App Resources" on page 1289.

Create Custom Apps from the Base App Package

The base MashZone NextGen app package includes simple stubs of all the required resources for an app.

1. Select **Visualize >** App Editor from the MashZone NextGen Hub menu.
2. Click **Create New App** in the Open App window to start a new app:
   a. Enter a **Name** for the app.

      Note: MashZone NextGen uses the app name to assign a unique identifier to the app. App names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

   b. Complete any of the optional meta-data for this app. See "Descriptions, Providers, Categories and Tags for Artifacts" on page 101 for more information.
   c. Click **Create New App**.

      The App Editor opens a new App Specification plus stubs for the other resources in the base app package.
3. Click **Save all and run** to test this app template.

   This initial template is a simple Hello World app that displays a message like this:

4. Use the pull-down list to open and update the HTML, JavaScript, CSS and App Spec files, as needed, to create the app. Add images or other CSS or JavaScript libraries as

needed. See the links to examples and techniques at the beginning of this topic for more information.

Files you add to the app are located within the folder tree for the app and must be referenced with relative URLs. Relative URLs *cannot* redirect to folders outside of the app's root folder.

Files can also be included using absolute URLs from external or internal sites.

You can also use any of the common JavaScript frameworks and plug-ins that are hosted in MashZone NextGen. Common examples include prototype, jQuery and many others. To see the list of hosted libraries, open the App Editor and click **Add**. You identify hosted libraries by name.

> **Note:** The libraries for Prototype and jQuery are *always* available for all apps.

5. Test the app using the **Save all and run** button.
6. Once the app is working, or at any time during development, click **Download** to zip up all the resources and folders in the app package and download this to your computer. You can then extract the files and folders to check this into your source control system.

Declare, Get and Set Properties in Custom Apps

In addition to the standard properties of all apps, such as width or title, any app can define additional properties in the `<properties>` section of its App Specification. For example:

```
...
<properties>
  <property name="customer" datatype="string"/>
  <property name="custNo" datatype="number" label="Customer Number"
    isinput="true" required="true"/>
  <property name="appID" datatype="string"
    defaultvalue="akZR92Yng345ty" isinput="false" visibility="hidden"/>
</properties>
...
```

For details on the syntax to declare properties, see the "<properties> or <property>" on page 1310 topic.

Additional app properties most commonly represent input parameters for the app. But you can use properties to track any characteristic of an app that you need to save to or retrieve from an app.

App properties must have a name and a simple datatype such as string or number. Property names must be valid JavaScript names and cannot begin with `presto` (this namespace is reserved).

> **Note:** Properties can be complex objects *only* when they are used to define the message payload for topics that an app publishes or subscribes to. See "Wiring App Interactions" on page 1249 for more information on topics and inter-app communication.

In addition, properties have several optional attributes that you can use, such as `defaultvalue`, `required` or `tooltip`. Most of these optional attributes are used in MashZone NextGen when users edit app properties. For example, the label attribute is the property name that users see when they update app properties in Mashboard or the AppDepot.

These optional attributes generally do not affect how the app runs. For example, the App Framework does not check to make sure that required properties have values when an app is first constructed.

Custom apps can access any of the attributes of a property and can use this information to implement specific behavior in the app. Custom apps can also set or get property values using the .MashZone NextGenApp API.

Get Properties or Property Values

The default app package that you download from the App Editor includes a basic example of retrieving the value of an app property and rendering this in the app.

Note: For links to other custom app examples, click **More...** above.

Hello World defines the property in its App Specification and a default value for the property:

```
<?xml version="1.0" encoding="UTF-8"?>
<app id="hello-world" name="Hello World"
    jsclass="Sample.Hello"
    height="200" width="200"
    draggable="false" minimizable="false">
  <title>Hello World</title>
  <description>The Hello World App</description>
<properties>
<property name="helloString" datatype="string"
defaultvalue="Techies !" label="Hello String">
<description>Hello String</description>
</property>
  </properties>
  <requires>
    <require name="jquery" type="library"/>
    <require src="js/app.js" type="script"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

The Sample.HelloWorld class uses the `getPropertyValue(name)` method in the MashZone NextGen App API to retrieve the property's value and render this in the app:

```
Sample.Hello = function( app ) {
  jQuery( app.rootElement ).find( '.helloString' ).html(
app.getPropertyValue( 'helloString' )
  );
};
```

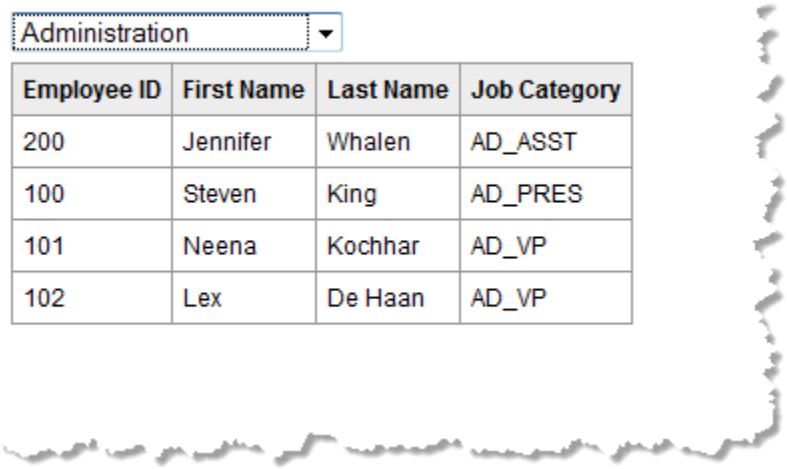
There are several other methods you can use in custom apps to retrieve or manipulate properties, such as `getProperty(name)` or `getPropertyNames()`. See the MashZone NextGen API Reference for details.

Set Property Values

This example is a simple custom app that runs a mashup to retrieve a set of employees and render them in a table. It is very similar to the custom table app shown in .

In this case, the app uses a job category that users choose to select employees. Initially, it opens with just a list of job categories.

When a user selects a job category, this value updates a property in the app. This is then used as an input parameter to the mashup to retrieve employee information and render the table.



The name of this sample custom app is `EmployeesByJobArea`. See "HTML and CSS for the Sample Employee App" on page 1230 for the basic structure and styles used in this custom app.

The App Specification has configuration to define the custom property used to track the current job category. It also adds the jQuery Templating Library which is used to render the app. See the "Properties and Libraries for the Sample Employee App Specification" on page 1230 section for details.

The remaining implementation details for this sample app are handled in the JavaScript library js/app.js. This includes:

- Create the App Constructor Using onLoad
- Create the jQuery Template for Table Rows

- Bind a Handler to the Job Category List and Update the App Property
- Invoke the Mashup and Render the Table

HTML and CSS for the Sample Employee App

The EmployeesByJobCategory/html/app.html file defines the HTML elements needed to render the app. It is wrapped by a `<div>` element with an ID assigned for easy access. It contains a `<select>` element to define the pull-down list of job categories and a table to render the employee information from the mashup results:

```
<div
>
  <div
>
    <select id="dept"
>
      <option value="none">Select a Job Category</option>
      <option value="AD">Administration</option>
      <option value="AC">Customer Service</option>
      <option value="FI">Finance</option>
      <option value="IT">Information Technology</option>
      <option value="MK">Marketing</option>
      <option value="PU">Purchasing</option>
      <option value="SA">Sales</option>
      <option value="SH">Shipping</option>
      <option value="ST">Warehouse</option>
    </select>
  </div>
  <table
>
    <thead>
      <tr>
        <th>Employee ID</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Job Category</th>
      </tr>
    </thead>
    <tbody
></tbody>
  </table>
</div>
```

The EmployeesByJobCategory/css/app.css file defines the CSS styles for the app:

```
div.sampleEmployees { font-family: Arial, Helvetica, sans-serif;
    font-size: 9pt; margin-top: 5px; margin-bottom: 5px; }
table.myTable {border: 1px solid #999999; border-collapse: collapse; }
table.myTable thead {background-color: #ededed;}
table.myTable th, table.myTable td {padding: 5px; border: 1px solid #999999;
    font-size: 9pt;}
```

Properties and Libraries for the Sample Employee App Specification

The EmployeesByJobCategory/app.xml file is the App Specification:

```
<?xml version="1.0" encoding="UTF-8"?>
<app id="EmployeesByJobArea" name="EmployeesByJobArea"
    jsclass="Sample.EmpsByJobArea"
    width="400" height="400" draggable="false" minimizable="false">
  <title>Employees by Job Area</title>
```

```
  <description>A sample App illustrating input parameters used as App properties and setting prop
  <properties>
<property datatype="string" label="Choose a job prefix" name="jobPrefix">
<description>Job prefix to retrieving employees</description>
</property>
  </properties>
  <presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
  <requires>
<require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

In addition to the name, title, description, ID, JavaScript class and device compatibility flags, two areas of the App Specification have configuration pertinent to this sample:

- *Required resources* in `<requires>` has the list of basic resources (the `<require>` declarations for js/app.js, html/app.html and css/app.css) for the app.

  It also has a declaration for the jQuery Templating library which is added by name (the `<require>` declaration for `jquery-tmpl`). You can also add resources for an app for any file in the app folders or using fully qualified URLs. For more information, see the "<requires> or <require>" on page 1313 topic.

- *Properties* has a declaration for `jobPrefix` which will track the job category that users choose in the app.

Create the App Constructor Using onLoad

The `EmployeesByJobCategory/js/app.js` file is the last resource needed for this custom app. It defines the constructor function and all of the behavior for the app.

It begins by defining the namespace for this app using the `Presto.namespace` function. Defining the class for an app in a separate namespace ensures there are no JavaScript conflicts with other apps or with the containing page where apps are deployed.

```
Presto.namespace("Sample");
Sample.EmpsByJobArea = function( app ) {
};
```

This is followed by the constructor method for the app. The App Framework uses this constructor to instantiate the app and passes a reference to the app to the constructor.

In this case, the constructor also makes use of the `onLoad(app)` method from the MashZone NextGen App API. `onLoad` is the load event handler for the App Framework. It is called once all resources for an app and the DOM have fully loaded and receives a reference to the app just as the app constructor does.

```
Presto.namespace("Sample");
Sample.EmpsByJobArea = function( app ) {
this.app = app;
var self = this;
  //constructor called when DOM and all resources are loaded
this.onLoad = function(app) {};
};
```

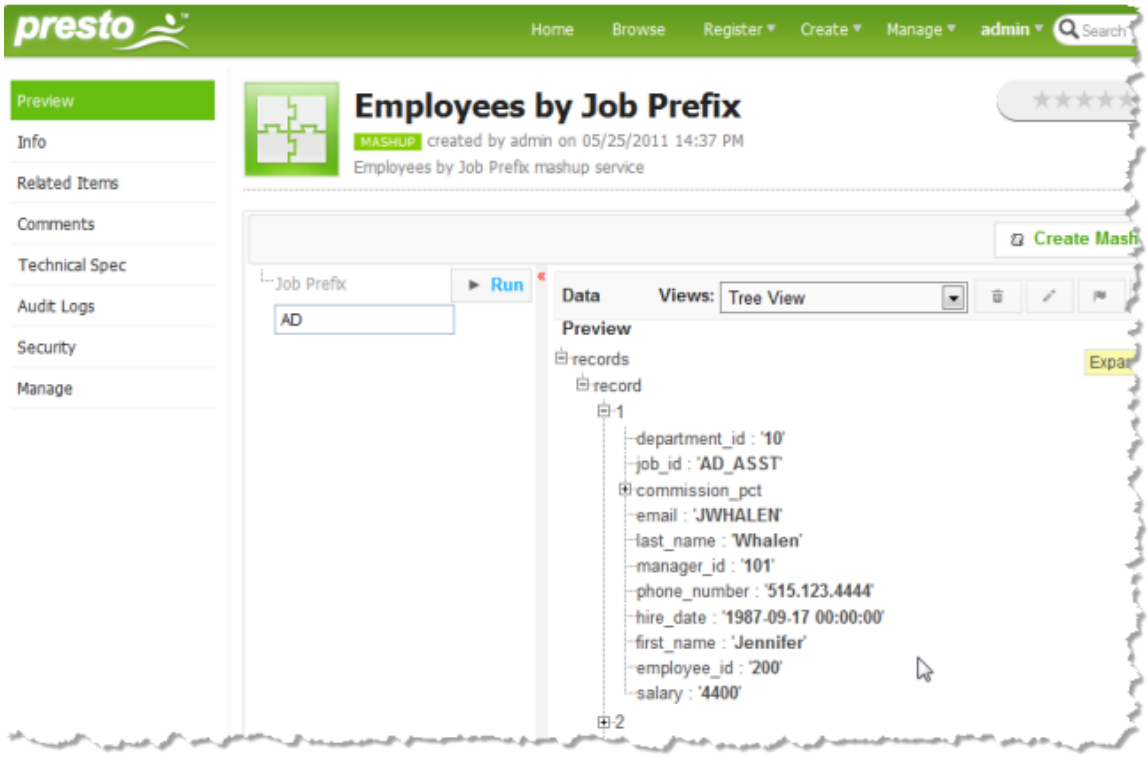
You can use `onLoad` to complete any initialization steps just as you do with the constructor. This example first gets a reference to the DOM node that wraps the app, using the `getRootElement()` method and also gets the default connection to the MashZone NextGen Server using `getConnection()`:

```
Presto.namespace("Sample");
Sample.EmpsByJobArea = function( app ) {
  this.app = app;
  var self = this;
  //constructor called when DOM and all resources are loaded
  this.onLoad = function(app) {
this.rootDiv = jQuery(this.app.getRootElement);
this.myTbl = this.rootDiv.find('.myTable');
this.dept = this.rootDiv.find('.deptOptions');
var curPrefix = this.dept.val();
    //hide table initially until job category selected
if (curPrefix = 'none') {
self.myTbl.hide();
} else {
self.getEmployees();
}
this.connection = this.app.getConnection();
this.requestBody = '';
  };
};
```

If there is no job category selected, this `onLoad` example also initially hides the table that will eventually display selected employee information.

Create the jQuery Template for Table Rows

The next step in the `onLoad` defines a jQuery template that will generate the HTML for each table row based on the results of the mashup. To create this template, you first need to see the actual structure of results for the mashable or mashup that will be populating data in the template. You can see this structure using the Tree View in the Preview tab of the artifact page for the mashup or mashable, such as this:



The mashup has a repeating `record` item that contains the employee fields that will populate the app's table. The template simply needs the appropriate HTML tags for one table row combined with the jQuery templating syntax to map to the appropriate result fields, such as this:

```
...
  //constructor called when DOM and all resources are loaded
  this.onLoad = function(app) {
    this.rootDiv = jQuery(this.app.getRootElement);
    this.myTbl = this.rootDiv.find('.myTable');
    this.dept = this.rootDiv.find('.deptOptions');
    var curPrefix = this.dept.val();
    //hide table initially until job category selected
    if (curPrefix = 'none') {
      self.myTbl.hide();
    } else {
      self.getEmployees();
    }
    this.connection = this.app.getConnection();
    this.requestBody = '';
var rowMarkup = "<tr><td>${employee_id}</td><td>${first_name}</td><td>${last_name}</td><td>${job_
jQuery.template("rowTemplate", rowMarkup);
   }
};
```

The `jQuery.template()` function compiles the template with `rowTemplate` as its name.

Bind a Handler to the Job Category List and Update the App Property

Next, the app constructor needs a handler to receive the event when users choose a job category. This function must:

- Set the jobPrefix property for the app
- Invoke the mashup, passing the appropriate job prefix as an input parameter
- Use the results to render the table with employee information

To bind the handler, the app class gets a reference to the `<select>` DOM node and binds the handler to the change event:

```
...
  this.onLoad = function(app) {
...
      jQuery.template("rowTemplate", rowMarkup);
      //bind handler for select
this.dept.change(function() {});
  }
};
```

To set the app property, the handler uses the `setProperty(`*`name`*`,`*`value`*`)` method:

```
...
  this.onLoad = function(app) {
...
      jQuery.template("rowTemplate", rowMarkup);
      //bind handler for select
      this.dept.change(function() {
var jobs = self.dept.val();
self.app.setPropertyValue('jobPrefix',jobs);
self.getEmployees();
        }
      );
  };
};
```

And then calls the `getEmployees` method to handle the invocation of the mashup and rendering the table.

Invoke the Mashup and Render the Table

The `getEmployees` method handles the last two tasks for the selection handler: invoke the mashup and render the table:

```
...
  this.onLoad = function(app) {
...
   };
//invokes mashup, renders table with employee results
this.getEmployees = function(){
var jobs = this.app.getPropertyValue('jobPrefix');
if (jobs != 'none') {
var prestoUrl = "/presto/edge/api/rest/Employees_by_Job_Prefix/runMashup?x-presto-resultFormat=js
this.connection.request({
url:  prestoUrl,
type: "get",
contentType: "application/x-www-form-urlencoded",
data: this.requestBody
```

```
},
{ onSuccess: function(response, responseHeaders) {
self.myTbl.show();
var result = response;
if (result.records.record) {
jQuery(".tblBody").empty();
var employees = result.records.record;
jQuery.tmpl("rowTemplate", employees).appendTo(".tblBody");
} else {
self.rootDiv.html("no results found");
}
},
onFailure: function(e) {
self.rootDiv.html(e.message);
}
});
} else {
self.myTbl.hide();
}
};
};
```

To invoke the mashup, the handler uses the default connection and the `request(reqConfig,callbacks)` method in PC4JS.

The `onSuccess` callback must use a successful response to render the table. It displays the table, removes any existing rows (from previous selections) and uses the jQuery template to render new rows based on the results.

The `onFailure` callback handles any error responses from invoking the mashup.

The complete JavaScript library is:

```
Presto.namespace("Sample");
Sample.EmpsByJobArea = function( app ) {
  this.app = app;
  var self = this;
  //constructor called when DOM and all resources are loaded
  this.onLoad = function(app) {
    this.rootDiv = jQuery(this.app.getRootElement);
    this.myTbl = this.rootDiv.find('.myTable');
    this.dept = this.rootDiv.find('.deptOptions');
    var curPrefix = this.dept.val();
    //hide table initially until job category selected
    if (curPrefix = 'none') {
      self.myTbl.hide();
    } else {
      self.getEmployees();
    }
    this.connection = this.app.getConnection();
    this.requestBody = '';
    var rowMarkup = "<tr><td>${employee_id}</td><td>${first_name}</td><td>${last_name}</td><td>${
    jQuery.template("rowTemplate", rowMarkup);
      //bind handler for select
      this.dept.change(function() {
          var jobs = self.dept.val();
          self.app.setPropertyValue('jobPrefix',jobs);
          self.getEmployees();
        }
      );
  };
//invokes mashup, renders table with employee results
```

```
  this.getEmployees = function(){
    var jobs = this.app.getPropertyValue('jobPrefix');
    if (jobs != 'none') {
        var prestoUrl = "/presto/edge/api/rest/Employees_by_Job_Prefix/runMashup?x-presto-resultF
        this.connection.request({
            url:  prestoUrl,
            type: "get",
            contentType: "application/x-www-form-urlencoded",
            data: this.requestBody
          },
          { onSuccess: function(response, responseHeaders) {
                self.myTbl.show();
                var result = response;
                if (result.records.record) {
                    jQuery(".tblBody").empty();
                    var employees = result.records.record;
                    jQuery.tmpl("rowTemplate", employees).appendTo(".tblBody");
                 } else {
                   self.rootDiv.html("no results found");
                 }
            },
            onFailure: function(e) {
              self.rootDiv.html(e.message);
            }
          });
    } else {
        self.myTbl.hide();
    }
  };
};
```

App Dimensions and Resizing

App dimensions and resizing needs depend on the device where the app is rendered:

- For mobile devices, apps always use the full dimensions of the device and orientation. Resizing due to orientation changes is handled automatically.
- For desktop devices, however, apps are published to a *container page* that can be in a variety of destinations. The container page may have several apps or other content that affect what space is available to the app in addition to the basic dimensions for the desktop device.

The *container element* determines the dimensions for the app and where it is rendered in the destination page. These dimensions must accommodate the app *plus* the elements provided by the App Framework shown above.

Note: Both destination containers or the app itself can hide the app title bar and either provide their own title bar or omit it entirely.

You can "Handle Resize Events" on page 1238 in the App Specification or in CSS. In many cases, you can simply let the browser reflow apps automatically when the

container is resized. You can also explicitly "Set App Dimensions" on page 1238 if needed.

Set App Dimensions

For mobile devices, the dimensions from the App Specification are ignored. The app is rendered in the full dimensions for that device and orientation.

In most desktop browser contexts, it is best to allow the container to control the overall dimensions for the app by setting the dimensions in the App Specification to `auto` (see "<app>" on page 1299 for details). This allows the app to fully fill the container element.

If you set specific dimensions in the App Specification and:

- The container dimensions are smaller, the app renders in the container with appropriate scroll bars.
- The container dimensions are larger, the app renders with the fixed dimensions from the App Specification rather than fully filling the container element.

Support Resizing in Custom Apps

In most cases, you can follow common web application techniques to allow app content to be resizeable in desktop devices, such as using percentages for dimensions and setting the CSS `overflow` property. Depending on the app's content, it is typically a good practice to add the following CSS properties to the root element for your custom app content:

- `overflow:auto`
- `height:100%`

In many cases, this is all that is required to allow custom app content to properly handle desktop dimension changes when the container is resized. If custom handling is required, however, you can explicitly "Handle Resize Events" on page 1238 in custom apps.

Handle Resize Events

Resize events affect both the app root node and the app content node (shown previously). If you need to explicitly handle desktop browser resize events for your custom app, you can either:

- Implement the `onResize(newWidth,newHeight)` method from the MashZone NextGenApp API in the JavaScript class for the app. This method allows you to handle resizing both the app content and the app title bar from the App Framework.

  For an example see:

  - "Example App for Custom Resizing" on page 1239
  - "Example 138. App Specification for Custom Resizing Example" on page 1240

- "Example 139. HTML for Custom Resizing Example" on page 1240
- "Example 140. CSS for Custom Resizing Example" on page 1240
- "Example 141. Example onResize Method" on page 1241

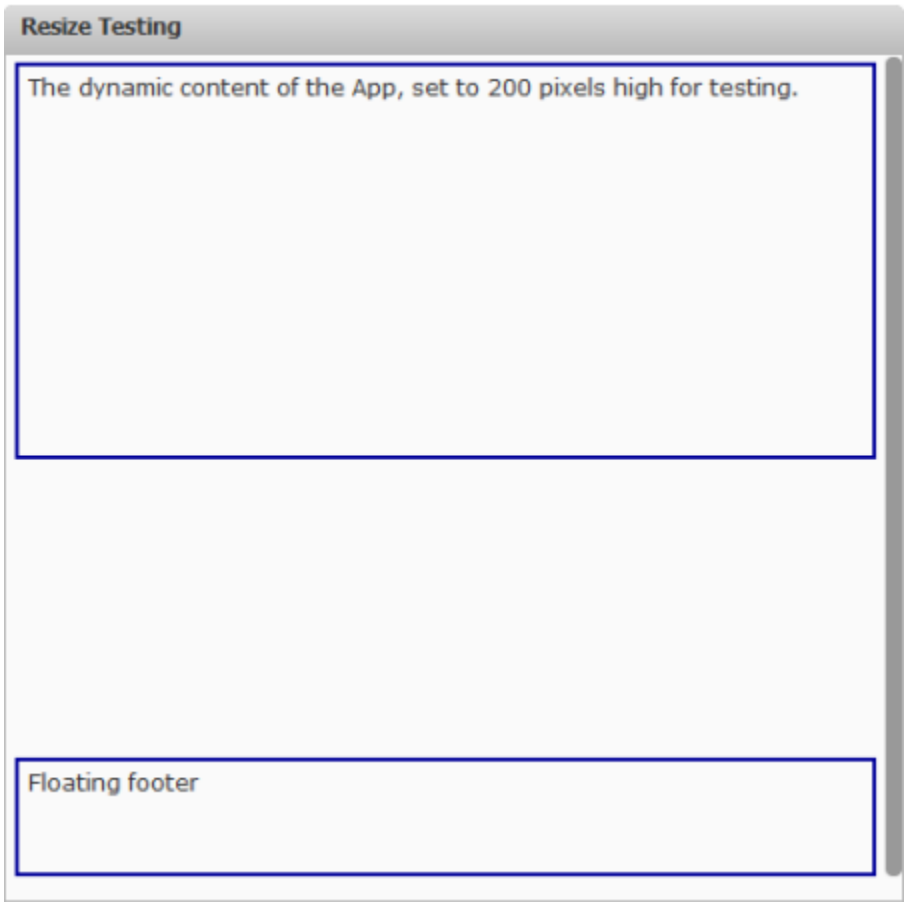
Note: For links to other app examples, click **More...** above.

- Add a listener for the resize event on the app content element and handle them there. This technique allows you to handle resizing for just the app content.

  For an example, see "Example 142. Example Resize Listener" on page 1242.

Example App for Custom Resizing

This example uses a mockup of an app, shown below, with two content areas: one for dynamic content and a second area for a footer that floats to the bottom of the app. The two content areas are mocked up with elements, literal text content and border styles to help visualize the impact of changes to the app's dimensions.



When resizing events happen, this example app must recalculate the top coordinate for the floating footer. The actual requirements and logic you need to handle resizing for your custom app will depend on its specific content and visual requirements.

App Specification for Custom Resizing Example

The App Specification for this example uses the base specification when you download an app package in the App Editor. It updates only those properties shown in bold:

```
<app id="resizeTest" name="Resize Test" jsclass="Sample.ResizeTest"
    draggable="false" minimizable="false" height="auto" width="auto">
  <title>Resize Testing</title>
  <description>Sample App to test resizing events from the container.</description>
  <properties/>
  <presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
  <requires>
    <require name="jquery" type="library"/>
    <require src="js/app.js" type="script"/>
    <require src="html/app.html" type="html"/>
    <require src="css/app.css" type="css"/>
  </requires>
</app>
```

Other than the basic updates to ID, name and class, this updates the width and height attributes. Note also that this app does not support mobile destinations.

HTML for Custom Resizing Example

The HTML for this example app is shown below:

```
<div
>
  <div
>
    The dynamic content of the App, set to 200 pixels high for testing.
  </div>
  <div
>Floating footer</div>
</div>
```

The content for each area is mocked up with simple text.

CSS for Custom Resizing Example

The CSS for this example app is shown below:

```
div.resizeContent {font-family: Verdana; overflow:auto; padding:5px;
height:95%; }
div.appContent {border: solid #000099 2px; padding:5px; height:200px; }
.floatingFooter {position:relative; left:0; border:2px solid #000099;
    padding:5px; height:50px; margin-top:5px;}
```

The height for the dynamic content area in this example is set to a specific size in the CSS for demonstration purposes. Once the app has been updated to retrieve dynamic content, the height property can be removed or reset to `auto`.

Relative positioning is used for the floating footer content. The left coordinate is set, but the top coordinate will be set in the `onResize` method within the app's class.

Also note the overflow and height properties for the `resizeContent` class which are applied to the root node for the app's content.

Example onResize Method

We start with the basics for the app class: defining a namespace and starting the constructor function for the app.

```
Presto.namespace('Sample');
Sample.ResizeTest = function( app ) {
  this.rootDiv = jQuery( app.getRootElement() );
  this.app = app;
    //minimum dimension to a minimum height for content and no footer overlap
  this.minH = 300;
};
```

The constructor defines two properties for the:

- Root node of the app (framework plus app content)
- Minimum height for the app (framework plus app content)

The minimum height property will be used to ensure a minimum height for the dynamic content area in the app. If the container height for the app is less than this minimum, the position of the floating footer is set based on this minimum height causing a vertical scroll bar to be added based on the `overflow` property assigned to the app's root content node.

Next we implement the `onResize` method. To ensure a minimum height for the app's dynamic content, we first must determine whether the position for the footer should be based on the current or minimum height:

```
Presto.namespace('Sample');
Sample.ResizeTest = function( app ) {
  this.rootDiv = jQuery( app.getRootElement() );
  this.app = app;
  //minimum dimension to a minimum height for content and no footer overlap
  this.minH = 300;
  this.onResize = function(curWidth, curHeight) {
    console.log('current height ' + curHeight );
    var footerTop;
    //choose current height or minimum height
    if (curHeight > this.minH) {
      footerTop = curHeight - this.minH;
    } else {
      footerTop = 0;
    }
    //set footer top position
    console.log('footer top ' + footerTop );
    this.rootDiv.find('.floatingFooter').css('top',footerTop);
  };
};
```

The final step in `onResize` is to set the footer top position.

Once `onResize` is implemented, the last step in the app's class is to handle the floating footer's position when the app is first rendered:

```
Presto.namespace('Sample');
Sample.ResizeTest = function( app ) {
  this.rootDiv = jQuery( app.getRootElement() );
  this.app = app;
  //minimum dimension to a minimum height for content and no footer overlap
  this.minH = 300;
```

```
  this.onResize = function(curWidth, curHeight) {
    console.log('current height ' + curHeight );
    var footerTop;
    //choose current height or minimum height
    if (curHeight > this.minH) {
      footerTop = curHeight - this.minH;
    } else {
      footerTop = 0;
    }
    //set footer top position
    console.log('footer top ' + footerTop );
    this.rootDiv.find('.floatingFooter').css('top',footerTop);
  };
  //set initial footer position
  var initWidth = this.app.getWidth();
  var initHeight = this.app.getHeight();
  this.onResize(initWidth, initHeight);
};
```

This uses the `getWidth` and `getHeight` methods from the MashZone NextGen API to get the initial dimensions and then calls `onResize` to set the footer position.

Example Resize Listener

This example uses the same App Specification, HTML and CSS as the previous example. See the following sections for details:

- "Example App for Custom Resizing" on page 1239
- "Example 138. App Specification for Custom Resizing Example" on page 1240
- "Example 139. HTML for Custom Resizing Example" on page 1240
- "Example 140. CSS for Custom Resizing Example" on page 1240

The app class for this example handles resize events by registering a listener for the resize event, rather than implementing the `onResize` method. The app class first gets a reference to the root node for the app and a minimum height for the app.

```
Sample.ResizeTest = function( app ) {
  this.app = app;
  var self = this;
  this.rootDiv = jQuery( app.getRootElement() );
  this.minH = 300;
var setFooter = function(event) {
console.log('current height ' + self.app.getHeight() );
var footerTop, curHeight = self.app.getHeight();
if (curHeight > self.minH) {
footerTop = curHeight - self.minH;
} else {
footerTop = 0;
}
console.log('footer top ' + footerTop );
self.rootDiv.find('.floatingFooter').css('top',footerTop);
  };
  //register listener on framework content root
jQuery(app.getContentEl()).resize(setFooter);
  //set initial footer positions
  setFooter();
};
```

The `setFooter` method in this class is a handler for the resize event. It determines the current height and then uses either the minimum height or the current height to set the top position for the floating footer in this example.

Next the app class registers a listener for the resize event passing `setFooter` as the handler function. This listener is registered on the app content element retrieved with the `getContentEl` method in the MashZone NextGen API.

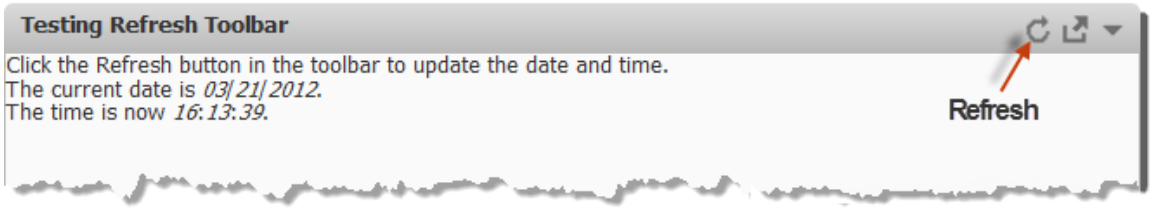
The final step is to call `setFooter` to set the position for the floating footer when the app is initially rendered.

Enable User-Initiated or Automatic Refreshes

Basic apps, either individually or within workspace apps, automatically provide several buttons in their title bar that allow users to update input parameters, open the app in a new window or refresh the app's data manually. Basic apps can also be configured to automatically refresh data periodically based on a *refresh interval*.

You can implement "User-Initiated App Refreshes" on page 1243 or "Automatic App Refreshes" on page 1246 for your custom app. In either case, MashZone NextGen will automatically include the **Refresh** toolbar button in the title bar for your custom app. See "Title Bars and Toolbar Buttons in Apps" on page 1112 for more information on this button.

The example in this topic presents a very simple app that invokes a mashup which returns the current date and time:



User-Initiated App Refreshes

To include the **Refresh** button in the default app Title Bar for user-initiated refreshes, a custom app must implement the `onRefresh` method from the MashZone NextGenApp API. See "The Structure of a MashZone NextGen App" on page 1290 for more information on the default app Title Bar.

The JavaScript class for this custom app that enables the Refresh button and handle refresh events is discussed in "Initial Render and Refresh Methods" on page 1244.

The additional resource files for this user-initiated example include: "HTML and CSS Source" on page 1245, the "App Specification for User-Initiated refreshTest" on page 1245 and the "Mashup EMML" on page 1245 that returns the current date and time.

Initial Render and Refresh Methods

The class for the refreshTest example app does most of the basics to declare a namespace, define a constructor for the class and invoke the refreshDateTime mashup to get the current date and time as components.

The requirements of rendering the current date and time, both initially and later when users click the Refresh button are found in the `onRefresh` and `getNow` methods:

```
Presto.namespace("Sample");
Sample.RefreshTest = function( app ) {
  this.rootDiv = jQuery(app.getRootElement);
  this.app = app;
  var connection = app.getConnection();
  var self = this;
  //define markup to render date and time as jQuery template
  var timeMarkup = "<p>Click the Refresh button in the toolbar to update the date and time. </p>"
    "<p>The current date is <span class='thismonth'>${month}</span>/<span class='thisday'>${day}<
    "<p>The time is now <span class='thishour'>${hour}</span>:<span class='thisminute'>${minute}<
  jQuery.template("timeTemplate", timeMarkup);
  //clear all exceptions and get current date and time
this.onRefresh = function() {
app.hideInfoMessage();
jQuery(".curtime").empty();
self.getNow();
};
  //get current date and time
this.getNow = function() {
connection.request({
url: "/presto/edge/api/rest/refreshDateTime/Invoke?x-presto-resultFormat=json",
type: "get",
contentType: "application/x-www-form-urlencoded",
data: ""
},
{ onSuccess: function(response, responseHeaders) {
if (response.result) {
var now = response.result;
jQuery.tmpl("timeTemplate", now).appendTo(".curtime");
} else {
self.rootDiv.html("no date or time to find!");
}
},
onFailure: function(e) {
app.handleException(e);
}
});
};
  this.getNow();
};
```

The `getNow` method handles the task of invoking the mashup to get the current date and time and using those results to render appropriate content in the app. This method is called at the end of the class to render data when the app is first loaded.

The `onRefresh` method clears any error messages that may have displayed, removes any existing data and then calls `getNow` to get update date and time information and render that.

HTML and CSS Source

The HTML and CSS source for this example are:

```
<div
></div>
```

and

```
div.curtime { display: block; padding: 5 10 8 10;}
div.curtime p {margin-top: 5; margin-bottom: 5;}
.thismonth, .thisyear, .thisday, .thishour, .thisminute, .thissec {
  font-style: italic;
}
```

App Specification for User-Initiated refreshTest

The properties that were updated specifically for user-initiated refreshes in this example are shown in bold:

```
<app id="refreshTest" jsclass="Sample.RefreshTest" name="refreshTest"
    minimizable="false" draggable="false" height="auto" width="auto">
<title>Testing Refresh Toolbar</title>
  <description>Custom App to test enabling the Refresh button.</description>
  <properties/>
  <presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
  <requires>
<require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

Mashup EMML

This mashup uses the standard XPath function, `current-dateTime()`, to get the current date and time and then parse this into individual components that are returned as a complex document. This result is then used in the example app to illustrate the implementation requirements for the app Refresh toolbar button.

```
<mashup name = "refreshDateTime"
    xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmashup.org/schemas/v1.0/EMML/
    ../schemas/EMMLPrestoSpec.xsd"
    xmlns="http://www.openmashup.org/schemas/v1.0/EMML"
    xmlns:macro="http://www.openmashup.org/schemas/v1.0/EMMLMacro">
  <output name="result" type="document"/>
  <variable name="nowString" type="string" />
  <assign fromexpr="current-dateTime()" outputvariable="$nowString"/>
  <variable name="year" type="string"/>
  <assign fromexpr="substring($nowString,1,4)" outputvariable="$year"/>
  <variable name="month" type="string"/>
  <assign fromexpr="substring($nowString,6,2)" outputvariable="$month"/>
  <variable name="day" type="string"/>
  <assign fromexpr="substring($nowString,9,2)" outputvariable="$day"/>
  <variable name="hour" type="string"/>
  <assign fromexpr="substring($nowString,12,2)" outputvariable="$hour"/>
  <variable name="minute" type="string"/>
```

```
  <assign fromexpr="substring($nowString,15,2)" outputvariable="$minute"/>
  <variable name="second" type="string"/>
  <assign fromexpr="substring($nowString,18,2)" outputvariable="$second"/>
  <constructor outputvariable="result">
    <result>
      <year>{$year}</year>
      <month>{$month}</month>
      <day>{$day}</day>
      <hour>{$hour}</hour>
      <minute>{$minute}</minute>
      <second>{$second}</second>
    </result>
  </constructor>
</mashup>
```

Automatic App Refreshes

The requirements to support automatic refreshes for a custom app are identical to the requiredments for user-initiated refreshes with one exception:

- The custom app class must implement the `onRefresh()` method just the same as for user-initiated refreshes. For this example, see "Initial Render and Refresh Methods" on page 1244.
- The HTML and CSS has no changes. See "HTML and CSS Source" on page 1245 for this example.
- The underlying mashup for this example, shown in "Mashup EMML" on page 1245, is identical.
- The App Specification, however, *must* identify the time interval to use for automatic refreshes in the `refreshinterval` attribute on the "<app>" on page 1299 element. This attribute identifies the time interval as a number of seconds which must be at least 5 seconds.

  See "App Specification for Automatic Refreshes" on page 1246 for an example.

App Specification for Automatic Refreshes

The properties that were updated specifically for automatic refreshes in this example are shown in bold:

```
<app id="refreshTest" jsclass="Sample.RefreshTest" name="refreshTest"
    minimizable="false" draggable="false" height="auto" width="auto"
refreshinterval="10" >
<title>Testing Automatic Refreshes</title>
  <description>Custom App to test enabling the Refresh button.</description>
  <properties/>
  <presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
  <requires>
    <require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

Handle Exceptions

Exceptions that apps may receive from the MashZone NextGen Server occur most commonly when apps are loaded or when they invoke mashups or mashables. Exceptions are typically handled in the `onFailure` callback for a request.

Some of the exceptions that MashZone NextGen can return include:

Some Types of Exceptions	Callback
Authentication or session timeout errors	onFailure
Authorization errors	
Missing dependencies (other artifacts or macros)	
Mashable availability errors because of authentication, authorization, network errors and so on	
Errors, such as invalid parameters, from a mashable. MashZone NextGen considers this case to be a success as the mashable information source typically returns a response to MashZone NextGen. Content within the response indicates that an error has occurred, but MashZone NextGen simply returns the response to the app. Apps may need to check for the presence of error messages in successful responses to properly handle these types of errors.	onSuccess
Missing resources for the app. Some types of missing resources, such as CSS or images, do not throw an exception, but instead allow the app to render as much as possible. For missing HTML or JavaScript or some other types of resources, however, the app generally doesn't render at all and the error is logged.	

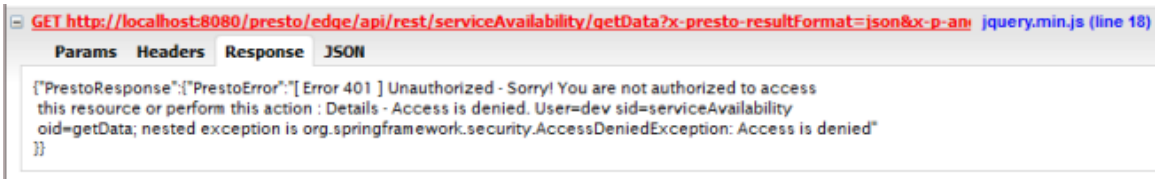
Exception Responses

If the app uses the default connection from the MashZone NextGen App API to connect to the MashZone NextGen Server, a MashZone NextGen exception response contains a single error object with the following properties:

- *message* = the error message for this exception.

- *code* = a number for an internal error code. The majority of exceptions do not have an error code, so this is most commonly zero.

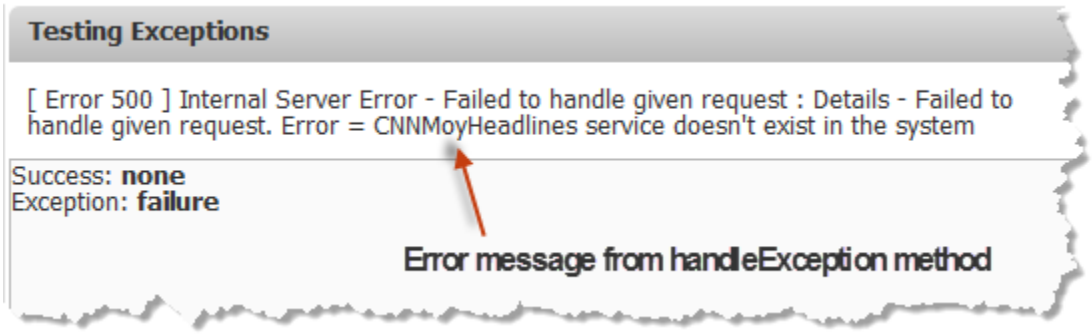
If the app uses another AJAX library to connect to the MashZone NextGen Server, the response for exceptions is a single object with a `PrestoResponse` property which contains a `PrestoError` property with the error message for this exception. For example:



MashZone NextGen App API Methods for Handling Exceptions

The MashZone NextGenApp API has convenience methods that you can use to handle exceptions, if your app uses the default connection to the MashZone NextGen Server:

- `handleException(e)`: displays the error message from the exception response in the default app Messages element provided by the App Framework. This appears just below the app title bar:



  For more information on the default app Message area, see "The Structure of a MashZone NextGen App" on page 1290.

- `hideInfoMessage()`: hides the default app Message area shown with `handleException` or `showInfoMessage`.

- `showInfoMessage(msg)`: identical to the `handleException` method, but the parameter to pass is just the message string, not the entire exception object.

The following example shows the app class for the Testing Exceptions app shown previously:

```
Presto.namespace("Sample");
Sample.ExceptionTest = function( app ) {
  var rootDiv = jQuery(app.getRootElement);
  var connection = app.getConnection();
  //clear any exception messages if refreshed
this.onRefresh = function(){
app.hideInfoMessage();
};
```

```
  //invoke service and update properties for which callback was used
  connection.request({
    url: "/presto/edge/api/rest/svcAvail/getData?x-presto-resultFormat=json",
    type: "get",
    contentType: "application/x-www-form-urlencoded",
    data: ""
    },
    { onSuccess: function(response, responseHeaders) {
        app.setPropertyValue('mySuccess','success');
        app.setPropertyValue('myError','none');
        rootDiv.find('.exceptionMsg').html(app.getPropertyValue('myError'));
        rootDiv.find('.successMsg').html( app.getPropertyValue('mySuccess'));
      },
onFailure: function(e) {
app.handleException(e);
        app.setPropertyValue('mySuccess','none');
        app.setPropertyValue('myError','failure');
        rootDiv.find('.successMsg').html( app.getPropertyValue('mySuccess'));
        rootDiv.find('.exceptionMsg').html(app.getPropertyValue('myError'));
      }
    }
  );
};
```

The `onRefresh` method uses `hideInfoMessage` to clear any exception messages. The `onFailure` callback for the request that invokes a mashable uses `handleException` to display the exception message.

Note: For links to other app examples, click **More...** above.

Wiring App Interactions

App interactions happen when an event in one app is communicated to other apps in the same page or container, allowing the receiving apps to take appropriate actions based on that event. This is also known as *inter-app communication* or simply as *interaction wiring*.

Events are propagated asynchronously between apps using a publish/subscribe paradigm. Event messages belong to *topics* that correspond to specific types of events. Apps publish or subscribe to topics.

Events and Interactions

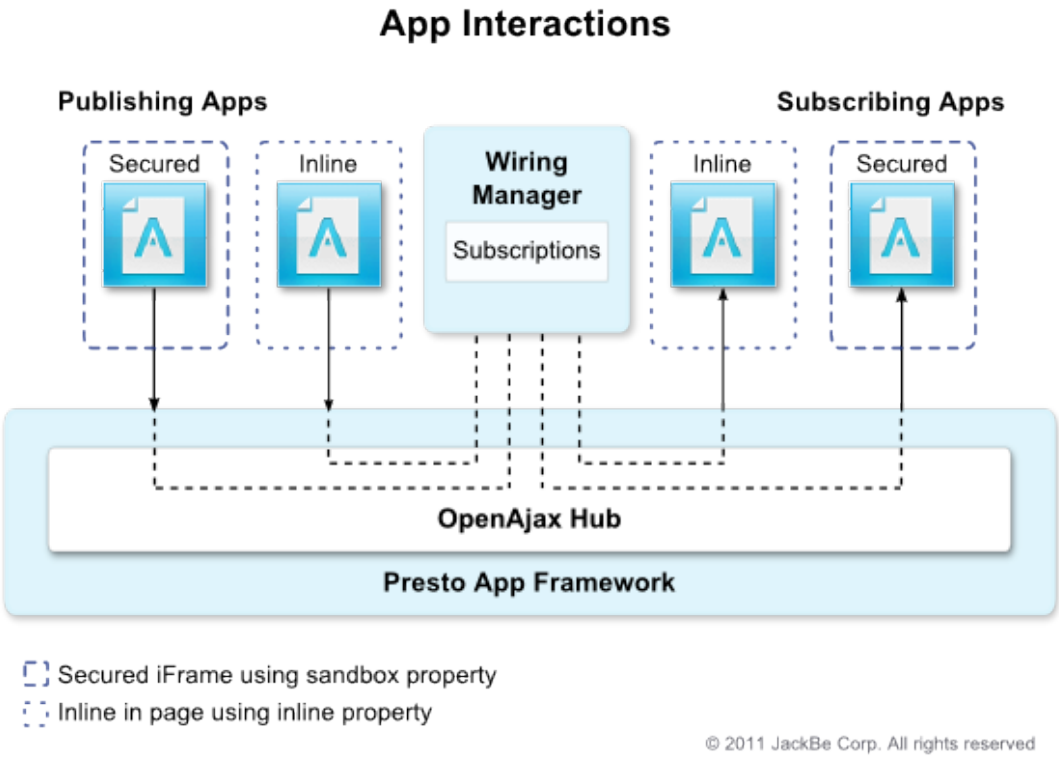
The types of events that can be published and the subscribing apps' interactions include:

	Publisher Events	Subscriber Behavior
Basic apps	The specific events supported by the views included in the app. For built-in MashZone NextGen views this typically includes user	Published message payloads can be used to either: ■ Update input parameters for the app to refresh information.

	Publisher Events	Subscriber Behavior
	click events for specific nodes within the view. See documentation for each built-in view for details. For pluggable views, the view declares the events it supports in the register method that adds the view to MashZone NextGen. See "Pluggable View Classes: Triggering and Handling Events" on page 1059 for more information. Built-in and pluggable views may also support row-select events from the underlying Datatable.	■ Determine row selection. The behavior or visual indications of row selection is specific to each view. One common behavior is to highlight one or more objects within the view.
Custom apps	Any browser event or custom event defined in the custom app.	Published messages can trigger any type of behavior defined in the subscribing custom app. Message payloads can be used to update app properties or determine row selection, but are not limited to these options.

Managing Interactions at Runtime

At runtime, the Wiring Manager in MashZone NextGen handles subscriptions and message transmissions for both secured and unsecured apps in a given page or container.



App communication is built on top of the OpenAjax Hub 2.0 Specification which provides the underlying functionality for securely passing event data between apps. App interactions are implemented using publish and subscription methods from the MashZone NextGen App API, plus optional configuration in the App Specification to support interactions that are loosely coupled.

Interaction Methods in the MashZone NextGen App API

For custom apps, there are three methods you can use to implement interactions:

- `publish([topic],message)`: to publish information about an event to any other apps that have subscribed to that topic. This method is used for both *tightly-coupled* and *loosely-coupled* interactions.
- `receive([topic],onMessage,scope)`: to receive messages for topics that other apps have published in a *loosely-coupled* interaction.
- `subscribe(topic,onMessage,scope)`: to subscribe to a known topic that is published from another app in a *tightly-coupled* interaction.

For detailed information about these methods, see the MashZone NextGenApp API.

The choice of method and any optional configuration needed to implement an interaction depends on whether the interaction is tightly-coupled or loosely-coupled.

Tightly-Coupled and Loosely-Coupled Interactions

Tightly-Coupled Interactions

With tightly-coupled interactions, both the publishing apps and the subscribing apps are designed to work together. All of the apps know the names of the topics that are published and the topic payloads (the event data). Each app implements an interaction using just the `publish` or `subscribe` API methods. The wiring between publisher and subscriber for each payload is implicit based on matching topic and property names.

Note: This represents the most basic form of app interaction. Tightly-coupled interactions were also supported inMashZone NextGen2.7 or earlier releases for mashlets.

With tight-coupling, all the apps in an interaction are typically designed and created at the same time. In many cases, the apps are not designed to work effectively without the other apps in the interaction.

A simple example of tight-coupling is a publisher app with a summary view of data combined with a subscribing app that has a detailed view of data, a *master/detail* relationship.

Loosely-Coupled Interactions

With loosely-coupled interactions, apps are designed to support interactions, but users choose which apps and interactions to actually wire to work together. Apps that support loose-coupling can be used alone and can also be wired with many different apps for different purposes. Because the apps that may be wired in an interaction are not necessarily created together with a specific interaction in mind, loose coupling is much more flexible.

For loose-coupling to work, apps must explicitly declare what topics they publish or subscribe to in their App Specification. Apps also must use the `publish` or `receive` API methods to support interactions.

Note: Basic apps that users create in the App Maker wizard support loosely-coupled interactions. They all subscribe to one topic that can be used in wiring interactions. If the views included in the basic app have events, the apps also publish topics.

Users can add apps that support loose coupling to a workspace in Mashboard and then explicitly wire the apps to complete the interaction configuration. Wiring allows the workspace to map mismatched topic names, and in many cases to map published topic payloads to the subscriber's topic payload.

Implementation Requirements

The types of apps and implementation requirements for tightly- or loosely-coupled interactions include:

	Tightly Coupled Interactions	Loosely Coupled Interactions
Basic apps	---	■ Use the `publish` and `receive` methods. ■ Require topic declarations in the App Specification. See "Declare App Topics and Payloads" on page 1285. ■ Must be wired in a workspace app by users in Mashboard. See "Enable Loosely-Coupled Interactions in Custom Apps" on page 1264 for details and examples.
Custom apps	■ Use the `publish` and `subscribe` methods. ■ Require specific parameters in the `<script>` tag that embed both publisher and subscriber apps. ■ Can subscribe to a set of topics using wildcards. See "Wildcard Subscriptions for Tightly-Coupled Interactions" on page 1262 for details. See "Tightly Coupled Interaction for Custom Apps" on page 1253 for details and examples.	

Tightly Coupled Interaction for Custom Apps

With tightly-coupled interactions, both publisher and subscriber apps know the topic name involved in an interaction and the topic payload for that type of event. Generally, both apps must be developed together to sustain tightly-coupled interactions.

In most cases, the apps involved in a tightly-coupled interaction are all custom apps. Basic apps created in the App Maker wizard can be used as publishers in a tightly-coupled interaction, but they cannot be subscribers.

There are two requirements to implement a tightly-coupled interaction:

- The `publish` and `subscribe` API methods. See "Publisher App for Tightly Coupled Interactions" on page 1254 and "Subscriber App for Tightly Coupled Interactions" on page 1257 for instructions and examples.
- Bundle the tightly-coupled apps together for end use in either:
  - A workspace app created in Mashboard.

    Simply create a new workspace in Mashboard (see "Create Workspace Apps with Mashboard" on page 1116 for instructions). Then add the tightly-coupled apps to the workspace or to one tab or page of a multi-tab or multi-page workspace.

No wiring is needed to enabled interactions between tightly-coupled apps that are bundled within a workspace app.

Save the workspace app. You can then publish the workspace to the AppDepot or publish or embed the workspace in other destinations to allow you and other users to use the tightly-coupled apps.

- External HTML pages.

  You can simply directly embed each app involved in a tightly-coupled interaction in any HTML page using `<script>` tags. You must use specific parameters in the `<script>` tags in this case. See "Embedding Tightly Coupled Apps" on page 1260 for details.

Subscribers can also use "Wildcard Subscriptions for Tightly-Coupled Interactions" on page 1262 to subscribe to a set of topics.

Note: For links to other app examples, click **More...** above.

Publisher App for Tightly Coupled Interactions

The example app used in this section extends the example app created in the "Declare, Get and Set Properties in Custom Apps" on page 1227 topic. This app uses the job category that a user selects to retrieve information on employees and display this data in a table.

To use this with another app that retrieves employee job history, this example adds a click event to the employee table to allow users to select an employee and then publishes this event to the tightly-coupled subscriber app.

Updates to HTML and CSS

This example adds a title to the HTML for the app:

```
<div
>
  <div
>Tightly Coupled Publisher</div>
  <div
>
...
  </div>
  <table
>
    <thead>
      <tr>
        <th>Employee ID</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Job Category</th>
      </tr>
    </thead>
    <tbody
></tbody>
  </table>
</div>
```

And adds a CSS style for the title:

```
div.sampleTightPub { font-family: Arial, Helvetica, sans-serif;
    font-size: 9pt; margin-top: 5px; margin-bottom: 5px; }
div.sampleTightPub select {margin-top: 5px; margin-bottom: 5px;}
div.title {font-size: 14pt; font-weight: bold; color: #555555;}
table.pubTable {border: 1px solid #999999; border-collapse: collapse; }
table.pubTable thead {background-color: #ededed;}
table.pubTable th, table.pubTable td {padding: 5px; border: 1px solid #999999;
    font-size: 9pt;}
```

Update Constructor Properties

Both the subscriber and publisher apps in this example are extensions of the example app created in the "Declare, Get and Set Properties in Custom Apps" on page 1227. To ensure that properties in both app are unique within the Sample namespace, we need to update the property name for the table node from `myTbl` to `pubTbl`:

```
Presto.namespace("Sample");
  Sample.TightCouplingPublisher = function( app ) {
  this.app = app;
  this.rootDiv = jQuery(app.getRootElement());
this.pubTbl = this.rootDiv.find('.pubTable');
  var self = this;
  //constructor called when DOM and all resources are loaded
  this.onLoad = function(app) {
    this.dept = this.rootDiv.find('.deptOptions');
    var curPrefix = this.dept.val();
    //hide table initially until job category selected
    if (curPrefix = 'none') {
      self.pubTbl.hide();
    } else {
      self.getEmployees();
    }
    ...
  };
};
```

Updating the Row Template to Support a Click Event

The table rows are generated in a jQuery template. To easily add a click event to each cell in these rows, this example adds a class to each table cell in the template HTML like this:

```
...
  this.onLoad = function(app) {
      ...
var rowMarkup = "<tr
>" +
"<td class='empId'>${employee_id}</td>" +
"<td class='fname'>${first_name}</td>" +
"<td class='lname'>${last_name}</td>" +
"<td class='jobId'>${job_id}</td>" +
"</tr>";
      jQuery.template("rowTemplate", rowMarkup);
      ....
   };
...
```

Binding a Handler to the Table Cell Click Event

To publish which employee a user selects, this example binds a handler to the click event for each cell in the table body. The click event only occurs if the `jobPrefix` property is not set to its default value of `none`:

```
...
  this.getEmployees = function(){
var jobs = this.app.getPropertyValue('jobPrefix');
if (jobs != 'none') {
      //invoke mashable, render table, bind click event
} else {
self.pubTbl.hide();
    }
  };
};
```

The table rows that this handler must bind to are generated in the `onSuccess` callback when the mashable is invoked to retrieve employee data, so binding must occur after that:

```
...
  this.getEmployees = function(){
    var jobs = this.app.getPropertyValue('jobPrefix');
    if (jobs != 'none') {
var prestoUrl = "/presto/edge/api/rest/Employees_by_Job_Prefix/runMashup?x-presto-resultFormat=js
this.connection.request({
url:  prestoUrl,
type: "get",
contentType: "application/x-www-form-urlencoded",
data: this.requestBody
        },
        { onSuccess: function(response, responseHeaders) {
self.pubTbl.show();
var result = response;
if (result.records.record) {
jQuery(".pubBody").empty();
var employees = result.records.record;
jQuery.tmpl("rowTemplate", employees).appendTo(".pubBody");
              //bind handler to click event and publish topic
jQuery(".employee").click(function(eventObj){
var thisRow = jQuery(this);
var thisEmpId = thisRow.find(".empId").text();
var thisFname = thisRow.find(".fname").text();
var thisLname = thisRow.find(".lname").text();
var thisJobId = thisRow.find(".jobId").text();
self.app.publish("selectEmpl", {"empId": thisEmpId, "fName": thisFname, "lName": thisLname, "jobI
});
} else {
self.rootDiv.html("no results found");
}
},
onFailure: function(e) {
self.rootDiv.html(e.message);
}
});
    } else {
        self.pubTbl.hide();
    }
  };
```

```
};
```

The handler uses the data from the row to build the topic payload and then publishes the message.

Subscriber App for Tightly Coupled Interactions

For subscribers in a tightly-coupled interaction, the only requirement is to use the `subscribe` method to register the subscription to a topic. Typically, this is done in the constructor for the app. In addition to identifying the topic, the subscriber must also supply a handler function to receive messages for this topic. The handler then takes appropriate action for each message.

The example in this section receives data for an employee from published messages and uses this to invoke a mashup and retrieve employee history. The results are then displayed in a table.

The HTML and CSS for the Subscriber App

The HTML for this subscriber app contains a title, simple user instructions and elements where the employee name and job history will be rendered:

```
<div
>
  <div
>Tightly Coupled Subscriber</div>
    <p>To see an employee's job history, select a job category and employee from Tightly Coupled
  <div
>
    <span id="fname"></span> <span id="lname"></span>
  </div>
  <table
>
    <thead>
      <tr>
        <th>Dates</th>
        <th>Job Title</th>
      </tr>
    </thead>
    <tbody
></tbody>
  </table>
</div>
```

The CSS styles are:

```
div.sampleTightSub { font-family: Arial, Helvetica, sans-serif;
    font-size: 9pt; margin: 5px;}
div.title {font-size: 14pt; font-weight: bold; color: #555555;}
div.empName{ font-size: 12pt; font-weight: bold; color: #555555;}
table.subTable {border: 1px solid #999999; border-collapse: collapse; }
table.subTable thead {background-color: #ededed;}
table.subTable th, table.subTable td {padding: 5px; border: 1px solid #999999;
    font-size: 9pt;}
```

The Subscriber App Specification

The App Specification for the subscriber is very basic, adding only the jQuery Templating library and updating the device compatibility <presto-meta> flags.

```
<?xml version="1.0" encoding="UTF-8"?>
<app id="Tight_Coupling_Subscriber" name="Tight Coupling Subscriber"
jsclass="Sample.TightCouplingSubscriber"
     minimizable="false" draggable="false"
     width="400" height="400">
  <title>Suscriber in a Tight Coupling</title>
  <description>This App is meant to always be used in conjunction with the Tight Coupling Publish
  <properties></properties>
<presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
<presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
<presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
  <requires>
<require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

Construct the Subscriber App and Define a jQuery Template

The JavaScript library for the subscriber app, declares a namespace for the app and starts the constructor using the `onLoad` method to ensure that all libraries and DOM nodes are loaded.

```
Sample.TightCouplingSubscriber = function( app ) {
  this.app = app;
  var self = this;
  //constructor called when DOM and all resources are loaded
this.onLoad = function(app) {
    this.rootDiv = jQuery(app.getRootElement());
this.subTbl = this.rootDiv.find('.subTable');
this.subtitle = this.rootDiv.find('.empName');
    //hide table and employee name until job category selected
this.subTbl.hide();
this.subtitle.hide();
    this.connection = app.getConnection();
    this.requestBody = '';
var rowMarkup = "<tr><td>${dates}</td><td>${job_title}</td></tr>";
    jQuery.template("rowTemplate", rowMarkup);
```

As is common with app constructors, this gets a reference to the containing page node that wraps the app as well as nodes of interest within the app and a connection to the MashZone NextGen Server. For more information on basic requirement for app constructors, see "Create the App Constructor Using onLoad" on page 1231.

Next the constructor defines a jQuery template that will be used to render the body rows of the table once job history data has been retrieved.

Register the Subscription

The last step for the constructor is to register a subscription using the `subscribe` method for the `selectEmpl` event that the tight-coupled publisher app publishes.

```
...
  this.onLoad = function(app) {
    this.rootDiv = jQuery(app.getRootElement());
    this.subTbl = this.rootDiv.find('.subTable');
    this.subtitle = this.rootDiv.find('.empName');
    //hide table and employee name until job category selected
    this.subTbl.hide();
```

```
    this.subtitle.hide();
    this.connection = app.getConnection();
    this.requestBody = '';
    var rowMarkup = "<tr><td>${dates}</td><td>${job_title}</td></tr>";
    jQuery.template("rowTemplate", rowMarkup);
this.app.subscribe("selectEmpl", function(topic,msg){},
self);
```

The second parameter is the handler function that will receive asynchronous messages from the publisher app.

Subscription Handler

The subscription handler must receive messages with one row of data for an employee and use this to invoke a mashup to retrieve job history for this employee and then render the results in a table.

```
...
    this.app.subscribe("selectEmpl", function(topic,msg){
jQuery("#fname").html(msg.fName);
jQuery("#lname").html(msg.lName);
this.subtitle.show();
var prestoUrl = "/presto/edge/api/rest/Job_History_for_Employee/runMashup?x-presto-resultFormat=j
this.connection.request({
url:  prestoUrl,
type: "get",
contentType: "application/x-www-form-urlencoded",
data: this.requestBody
},
{ onSuccess: function(response, responseHeaders) {
self.subTbl.show();
var result = response;
if (result.records.record) {
jQuery(".subBody").empty();
var jobs = result.records.record;
jQuery.tmpl("rowTemplate", jobs).appendTo(".subBody");
} else {
self.rootDiv.html("no results found");
}
},
onFailure: function(e) {
self.rootDiv.html(e.message);
}
});
}, self);
  };
};
```

Because this is tightly-coupled, the handler function knows the names of all properties in the message. The handler uses the first and last name from the message to render the employee name and then builds a request to invoke the mashup using the default connection and the remaining properties in the message.

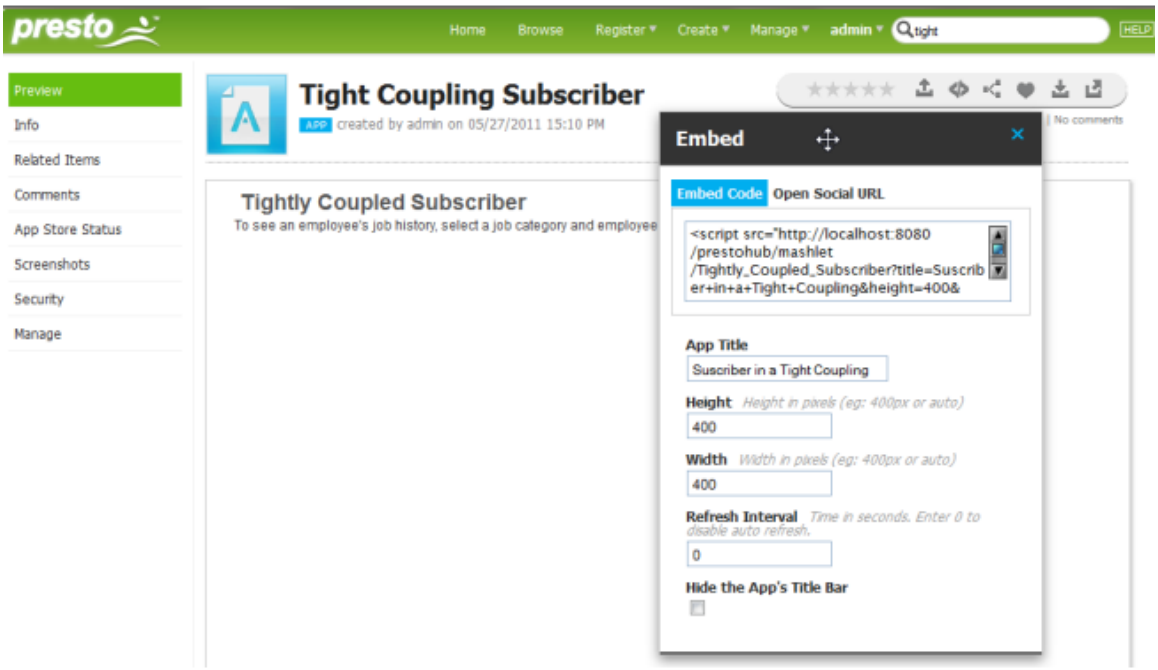
The request contains two callbacks: `onSuccess` and `onFailure`. The callback for successful responses uses the results to render the table with job history data. The callback for failure handles any errors.

Embedding Tightly Coupled Apps

Once you have implemented a tightly-coupled interaction in custom apps and uploaded them to MashZone NextGen, you are ready to bring the apps together. You can simply create a workspace app in Mashboard to hold the custom apps.

One other option is to embed the custom apps in a web page. First, you need the basic embed code for each app:

1. Open the app artifact page in MashZone NextGen and click **Embed**.
2. If needed, change any of the app properties and then copy the full <script> tag shown as Embed Code:



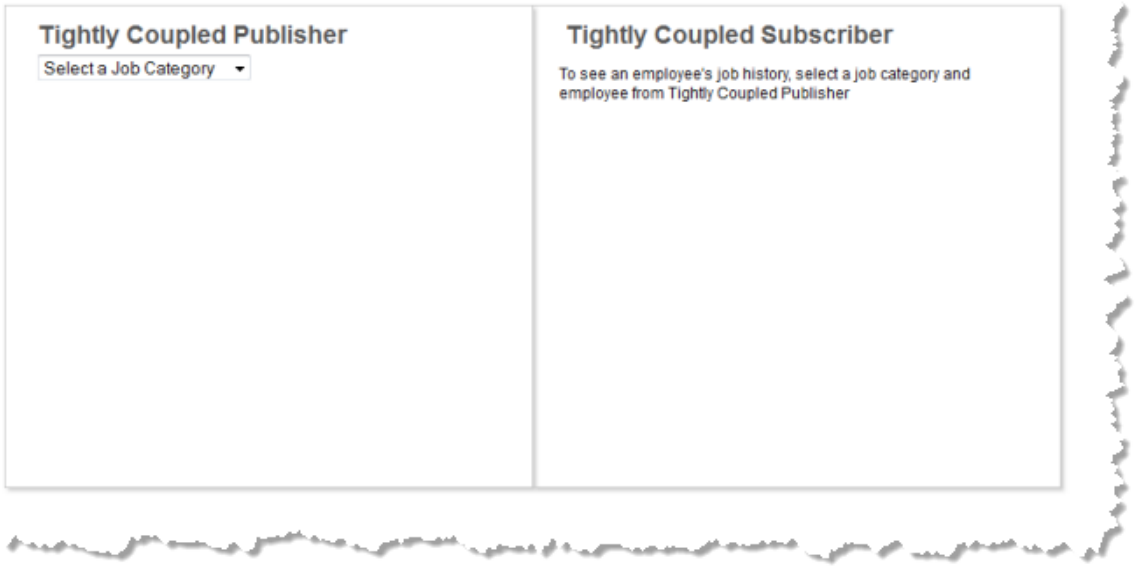
3. Paste this code into the web page where the app should appear. The or other element where you paste this <script> tag *must* have an ID assigned.
4. Then edit the parameters in the URL in the `src` attribute for the <script> tag:
   - Remove `&standalone=true` from the URL.
   - Get the ID of the element in the web page where this app will be rendered and add `&el=`*`parent-node-id`*.
   - Add `&inline=true` to the URL.

     This parameter allows the Wiring Manager and OA Hub to load in this web page before any apps that have interactions are loaded. It also removes the <iframe> which normally wraps each app.

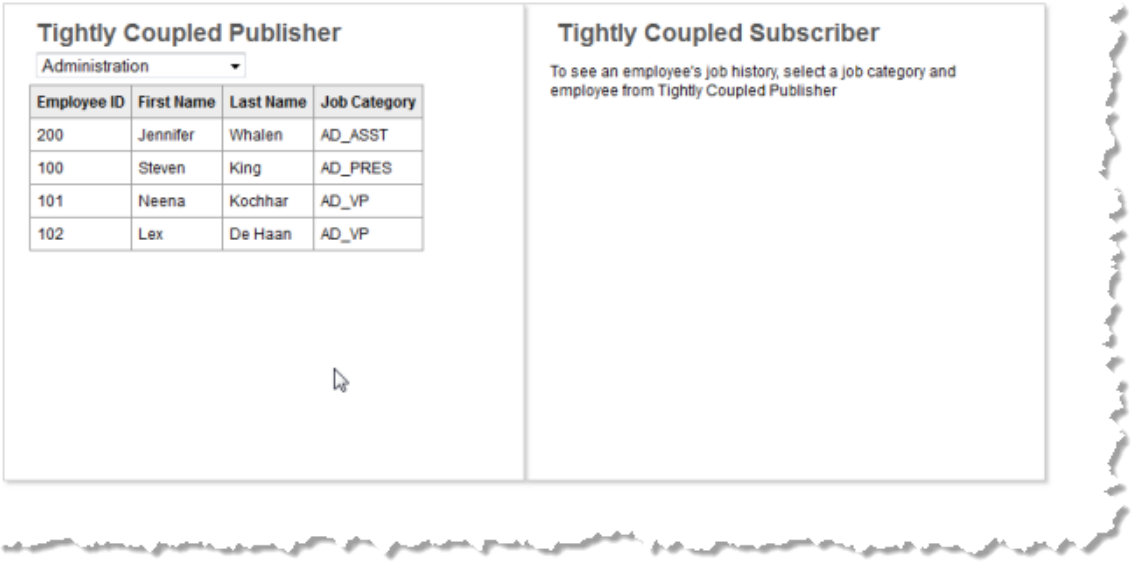
- If the app does need to be secured in an <iframe>, add `&sandbox=true`. This parameter, in conjunction with the `inline` parameter allows the Wiring Manager to communicate with apps involved in an interaction that are secured in an <iframe>.

Once you have embedded all the apps involved in the tightly-coupled interaction, test the web page and the interaction.

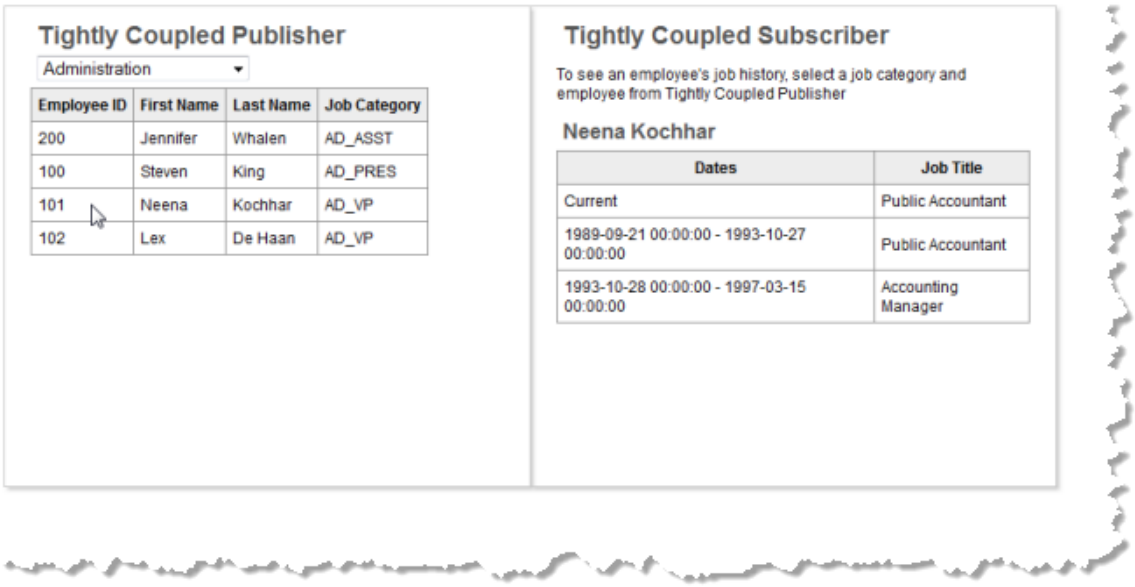
For the sample shown in this topic, the initial page renders both apps with no data:



Once a user selects a job category, basic employee data displays in the publisher app:



When a user selects an employee in the publisher app, this triggers the click event, publishing a message that is received by the subscriber app. The subscriber uses this message to retrieve job history and render that response:



Wildcard Subscriptions for Tightly-Coupled Interactions

Apps can subscribe to multiple topics that all belong to a category using wildcards in the topic name for the subscription. This is most useful, typically, in tightly-coupled interactions where topic names are well known and follow a pattern.

Note: You cannot use wildcard subscriptions with loosely-coupled interactions that are wired in Mashboard.

Wildcards can be used to represent any *token* within a topic name. This is based on the OpenAjax Hub 2.0 Specification definition that topic names are in the form:

token `.`*token* `.`*token...*

And only the first token is required.

You can use the asterisk `*` character as a wildcard for a given token within a topic name. Or use `**` as a wildcard for the last token and all subsequent tokens.

This example subscribes to any topic with a three-token name beginning with `division.` and ending with `.notice`, such as `division.east.notice` or `division.namerica.notice`:

```
...
  onLoad: function(app) {
    this.app = app;
    var self = this;
    ...
    app.subscribe("division.*.notice", function(topic,msg) {
```

```
      //subscription handler
      }, self);
    ...
  },
...
```

Or this example, which subscribes to any topic starting with `patient.`:

```
...
  onLoad: function(app) {
    this.app = app;
    var self = this;
    ...
    app.subscribe("patient.**", function(topic,msg) {
       //subscription handler
       }, self);
    ...
  },
...
```

Subscription Handlers for Wildcard Topic Names

A wildcard subscription provides a single handler for multiple subscriptions. If the apps are not secured, the handler can simply check the published topic name in each message to determine what to do with the message. For example:

```
...
    app.subscribe("patient.**", function(topic,msg) {
      if (topic === "patient.add") {
            addPatient(msg);
        } else if (topic === "patient.transfer") {
            transferPatient(msg);
        } else {
            console.error("unknown patient event");
        }
      }, self);
    ...
  },
...
```

If the apps are secured, however, the published topic name is generated during the process of tunnelling through the secured <iframe>, so topic names *will not* match. To overcome this, it is a better practice to check the message payload to determine which event a message represents. For example:

```
...
    app.subscribe("patient.**", function(topic,msg) {
      if (msg.pt.status === "new") {
            addPatient(msg);
        } else if (msg.pt.status === "transfer") {
            transferPatient(msg);
        } else {
            console.error("unknown patient event");
        }
      }, self);
    ...
  },
...
```

Enable Loosely-Coupled Interactions in Custom Apps

With loosely-coupled app interactions, publisher and subscriber apps:

- Can be created at different times by different users.
- Typically do *not* know the topic names or payloads for other apps involved in any interaction.
- Can be used in many different workspaces and many different interactions. In many cases, they can also be used without any other apps or interactions.
- Can be custom or basic apps. Both custom and basic apps can be involved in an interaction as publisher or subscriber or both.
- Users choose which apps to combine and wire together in a workspace using Mashboard.
- Wiring maps the mismatched topic names and payloads to allow the interaction work.

Custom apps that support loosely-coupled interactions must use the `publish(topic,msg)` or `receive(topic,msg)` API methods. See the MashZone NextGenApp API for detailed information on this API.

They must also declare the topics and payloads that the app publishes or subscribes to in their App Specification. For more information and samples of the tasks involved, see:

- "Declare App Topics and Payloads" on page 1285
- "Publisher App for Loosely-Coupled Interactions" on page 1264
- "Subscriber App for Loosely-Coupled Interactions" on page 1272
- "Basic App Support for Loosely-Coupled Interactions" on page 1282

This topic builds two sample custom apps named `Loosely Coupled Publisher` and `Loosely Coupled Subscriber`. It also uses a basic app created using a sample web service, Xignite Historical Quotes, that is shipped with MashZone NextGen.

Complete samples for the HTML, CSS and App Specifications are provided in the sections describing these sample apps. For complete samples of the JavaScript classes for these two custom apps, see the The Completed Sample Apps section.

Note: For links to other app examples, click **More...** above.

Publisher App for Loosely-Coupled Interactions

This section builds a sample custom app named `Loosely Coupled Publisher`. This custom app runs a mashup to combine customer and project information and produce a summary list of current customer projects.

`Loosely Coupled Publisher` publishes two topics when users select a customer project from the list by clicking any cell with customer data. The steps to create this sample app include:

- HTML and CSS for the Loosely-Coupled Publisher Sample
- Declaring Publish Topics in the App Specification
- The Initial Constructor for the Loosely-Coupled Publisher Sample
- Adding the Row Click Handler to Publish Messages
- Wiring a Published Topic in Mashboard

HTML and CSS for the Loosely-Coupled Publisher Sample

The HTML for `Loosely Coupled Publisher` has a simple heading and the rough outline for the table that will be populated with customer project information:

```
<div
>
  <h2>Loosely Coupled Publisher Sample</h2>
  <table
>
    <thead>
      <tr>
        <th>Company</th>
        <th>Symbol</th>
        <th>Project $</th>
        <th>Completion Date</th>
        <th>Location</th>
      </tr>
    </thead>
    <tbody
></tbody>
  </table>
</div>
```

The CSS is also very simple. It is *important*, however, that all of the styles are class-based to ensure that they do not affect any other apps that may be used in workspaces with this custom app.

```
div.loose-wiring-sample { font-family: Arial, Helvetica, sans-serif; }
div.loose-wiring-sample h2, div.loose-wiring-sample h3 { font-weight: bold;
    color: #555; padding-top: 10px; }
div.loose-wiring-sample div { font-size: 9pt; margin: 5px; }
table.projectTable { border: 1px solid #999999; border-collapse: collapse;
    width: 90%; }
table.projectTable thead { background-color: #ededed; font-size: 9pt; }
table.projectTable th, table.projectTable td { padding:5px;
     border: 1px solid #999999; }
```

Declaring Publish Topics in the App Specification

In addition to the basic requirements for an App Specification for a custom app, custom apps that support loosely-coupled interactions must declare the topics that the app publishes and the payload for messages published to that topic in their App Specification.

To declare topics, add a <topics> section, with a <topic> child for each topic that an app publishes or subscribes to:

```
<?xml version="1.0" encoding="UTF-8"?>
<app name="Loosely Coupled Publisher" id="Loosely_Coupled_Publisher"
jsclass="Sample.LooselyCoupledPublisher"
     minimizable="false" draggable="false" height="auto" width="600">
  <title>Loosely Coupled Publisher Sample</title>
  <description>Sample demonstrates an App that publishes two separate events for use in loosely-c
<dependson>
<resource name="CustomerProjectSummary" operation="runMashup"
type="service"/>
</dependson>
  <properties/>
  <presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
<topics>
<topic name="Sample.location" datatype="object" publish="true">
<description>Selected client location information</description>
<properties>
<property name="city" datatype="string"/>
<property name="state" datatype="string" />
<property name="zip" datatype="number"/>
</properties>
</topic>
<topic name="Sample.stock" datatype="object" publish="true">
<description>Selected client stock and project information</description>
<properties>
<property name="company" datatype="string" />
<property name="symbol" datatype="string"/>
</properties>
</topic>
</topics>
  <requires>
<require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

Each topic has a name and a datatype which can be either a simple type or `object`, for complex message payloads, or `any`. You define the payload of the messages that the app publishes in <properties> within <topic>. For more details on declaring topics and message payloads, see "Declare App Topics and Payloads" on page 1285.

This App Specification also includes the basic changes to the default specification, such as `id`, `name`, and `jsclass`, and has added the jQuery Templating library in `<requires>`. It also uses the `<dependson>` section to declare the dependency to the mashup that produces combined customer and project data and has the appropriate `<presto-meta>` device compatibility flags set.

The Initial Constructor for the Loosely-Coupled Publisher Sample

The constructor for this sample custom app handles the basic tasks:

- Declaring a namespace

- Getting the root node for the app
- Getting references to nodes of interest within the app
- Defining the jQuery template that will be used to generate table rows
- Connecting to MashZone NextGen and invoking the mashup to retrieve project information
- And finally rendering the data.

```
Presto.namespace("Sample");
  Sample.LooselyCoupledPublisher = function( app ) {
    var root = jQuery( app.getRootElement() ),
projectTable = root.find(".projectTable"),
projectBody = projectTable.find(".tblBody"),
rowMarkup = "<tr class='project'>"+
"<td class='company'>${customer_company}</td>"+
"<td class='symbol'>${customer_symbol}</td>"+
"<td class='projectValue' align='right'>${project_value}</td>"+
"<td class='projectEta'>${project_eta}</td>"+
"<td>"+
"<span class='city'>${customer_city}</span>, "+
"<span class='state'>${customer_state}</span> "+
"<span class='zip'>${customer_zip}</span>"+
"</td>"+
"</tr>";
    rowTemplate = jQuery.template("rowTemplate", rowMarkup);
    //invoke mashup
prestoUrl = "/presto/edge/api/rest/CustomerProjectSummary/runMashup?x-presto-resultFormat=json";
    this.onLoad = function(app) {
        app.getConnection().request({
            url: prestoUrl,
            type: "get",
            contentType: "application/x-www-form-urlencoded",
            data: ""
        }, {
onSuccess: function(response) {
if (response.result && response.result.customer_project) {
var projects = response.result.customer_project;
projectBody.empty();
jQuery.tmpl(rowTemplate, projects).appendTo(projectBody);
} else {
projectTable.hide();
}
},
onFailure: function(e) {
app.handleException({
message: 'Failed to retrieve project info: ' + e.message
});
}
        });
    };
};
```

Adding the Row Click Handler to Publish Messages

In addition to rendering mashup data, this custom app must publish messages to two topics when a user selects a project by clicking on a row in the table. To do this the app needs an event handler to build and publish the messages:

```
...
    rowTemplate = jQuery.template("rowTemplate", rowMarkup);
    //inner function to publish topics for row click
onRowClick = function(event) {
var row = jQuery(this),
customer = row.find("td.company").text(),
symbol = row.find("td.symbol").text(),
city = row.find("td span.city").text(),
state = row.find("td span.state").text(),
zip = row.find("td span.zip").text();
app.publish("Sample.stock", {
"company": customer,
"symbol": symbol
});
app.publish("Sample.location", {
"city": city,
"state": state,
"zip": zip
});
},
```

The event handler receives an event object for the user action, in this case the click event, and uses this to find the data needed to build each message and publish them using the `publish` method from the App API.

This handler must also be bound to each row of data after the rows have been generated in the onSuccess callback:

```
..
        app.getConnection().request({
            ...
        }, {
            onSuccess: function(response) {
                if (response.result && response.result.customer_project) {
                    var projects = response.result.customer_project;
                    projectBody.empty();
                    jQuery.tmpl(rowTemplate, projects).appendTo(projectBody);
//bind handler to row click event
projectBody.find("tr.project").click(onRowClick);
                } else {
                    projectTable.hide();
                }
            },
            onFailure: function(e) {
                app.handleException({
                    message: 'Failed to retrieve project info: ' + e.message
                });
            }
        });
...
```

Wiring a Published Topic in Mashboard

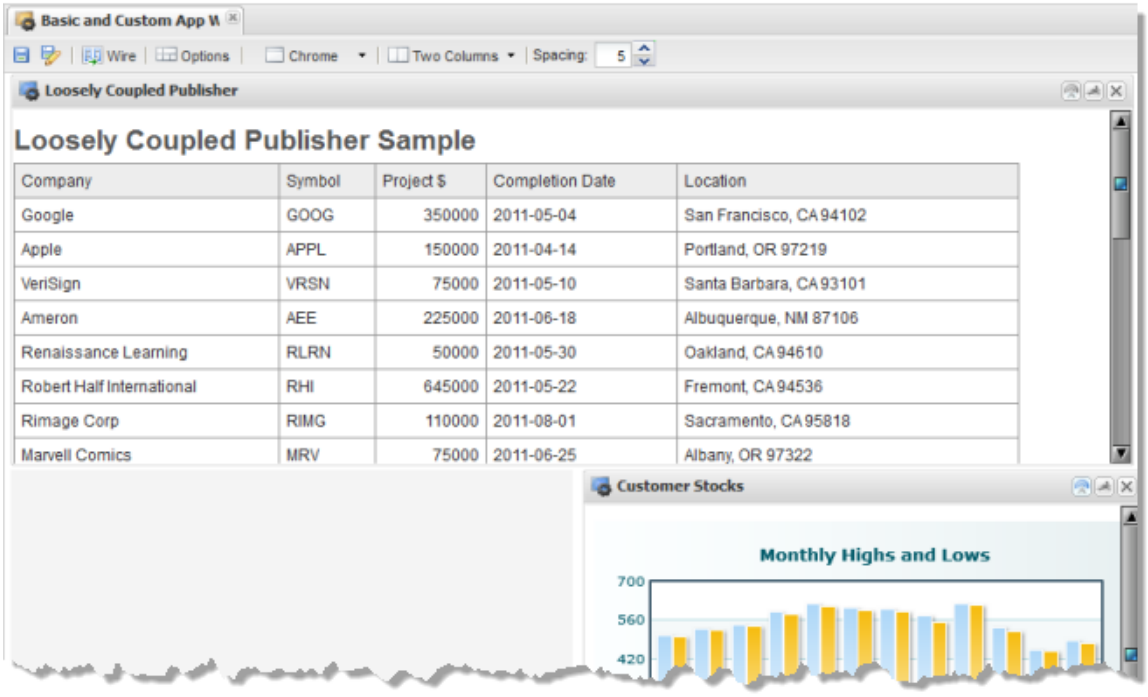
You can test the `Loosely Coupled Publisher` app using the index.html file that comes in the standard custom app template. This will invoke the mashup and render a table of project data. However, with loosely-coupled interactions, you cannot test the interaction until you have wired the app to another app in a workspace in Mashboard.

First, upload `Loosely Coupled Publisher` or your custom app using the App Editor (for more information, see "Create Custom Apps from the Base App Package" on page 1226). To test the interaction:

1. Select **Visualize >** Mashboard in the MashZone NextGen Hub main menu to open Mashboard.
2. Start a new workspace. For this sample, use a Table layout with one cell in the top row and two cell in the second row.
3. Drag `Loosely Coupled Publisher` or your custom app into the workspace. For this example, put `Loosely Coupled Publisher` in the top row of the workspace.
4. Drag another app that is appropriate for the interaction and that has at least one subscription topic.

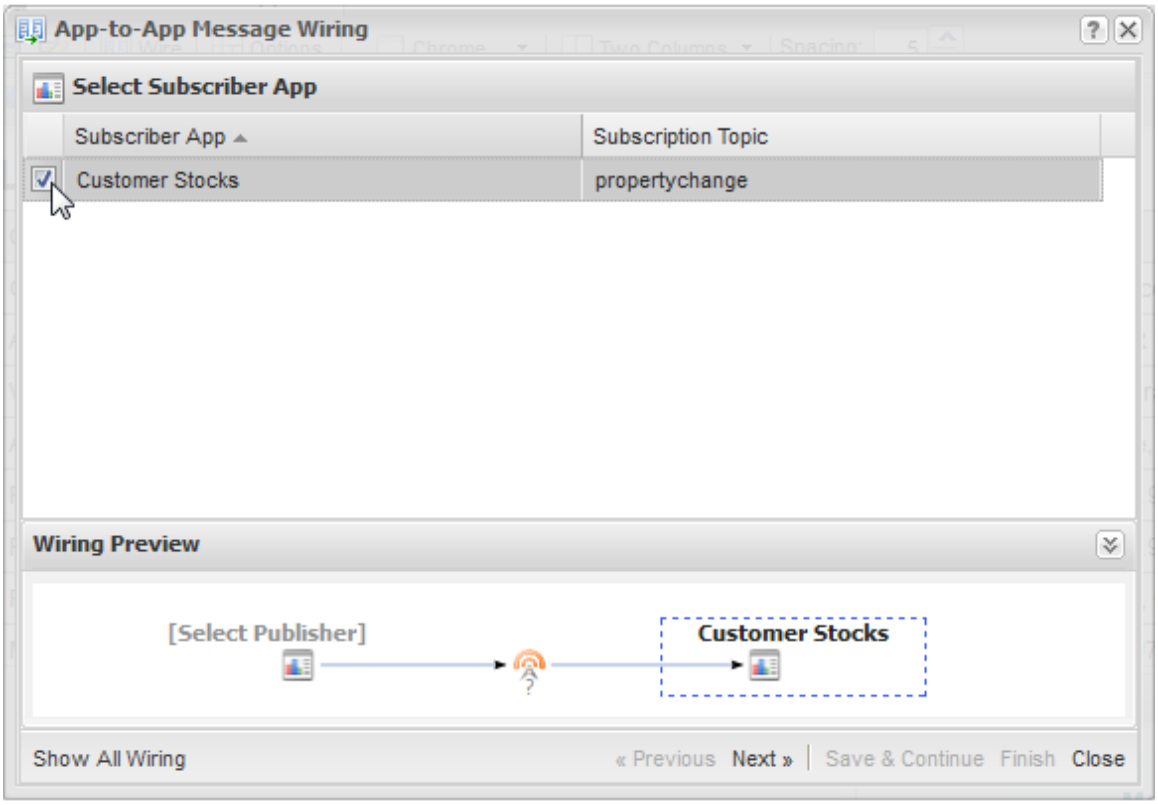
   Note: Basic apps that have input parameters automatically have one subscription topic, named `propertychange`.

   For this sample, we will use a basic app named `Customer Stocks` and drag in into a cell in the second row of the workspace.

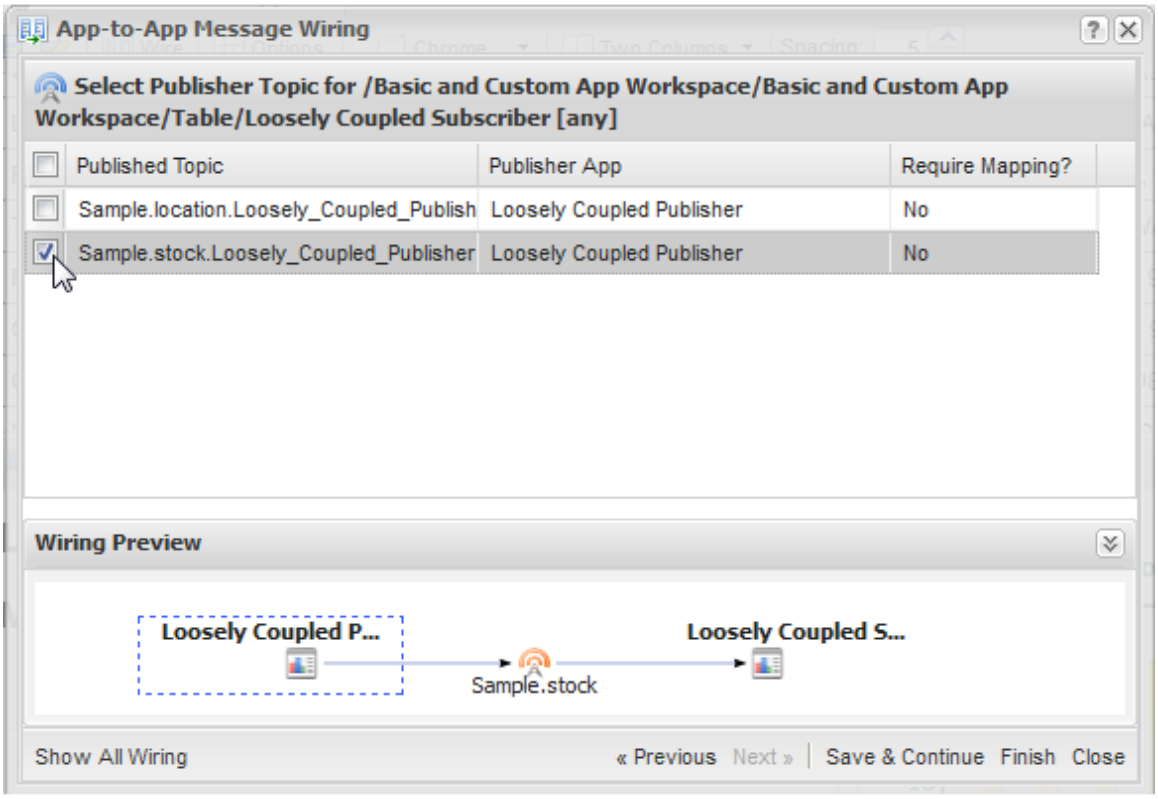


   `Customer Stocks` uses a mashup with an input parameter for one stock symbol to retrie[illegible]e high and low values for that stock which is shown in a colum[illegible]

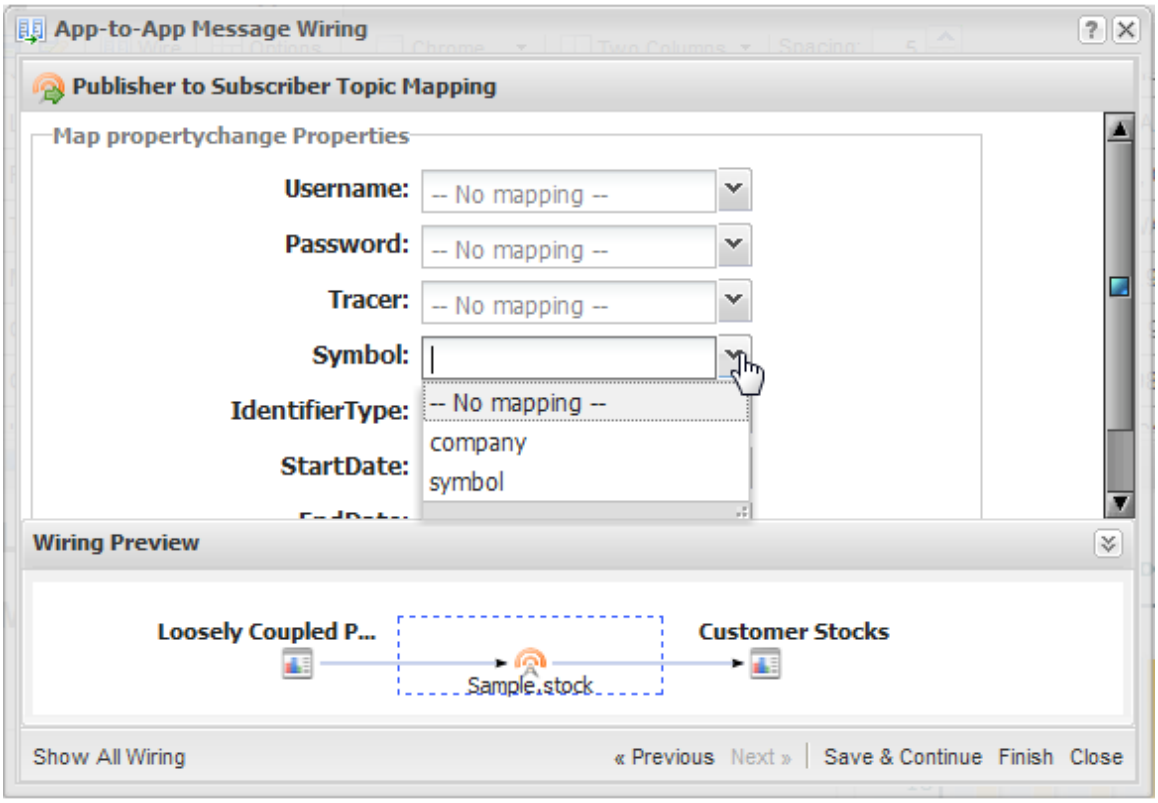
5. Click [illegible] **Wire** and choose the subscription topic that should receive a published message from `Loosely Coupled Publisher` or your custom app.



6. Click **Next** and choose the publish topic to wire to this subscription. In this sample, choose the `Sample.stock` topic from `Loosely Coupled Publisher`.



7. If the published message payload does not match the subscription message payload, click **Next** to map the appropriate message properties. In this sample, you must match `symbol` from the published message to `Symbol` in the subscription message:



8. Click **Finish**.
9. To test the interaction, perform the action in the publisher app that should publish a message. In this sample, simply click on one of the project rows in `Loosely Coupled Publisher`. The chart in `Customer Stocks` updates for each selection to show 12 months of stock for the selected customer.
10. Save this workspace as `Basic and Custom Apps Workspace`.

To test the other publish topic in `Loosely Coupled Publisher`, we first need to build a custom app that supports loosely-coupled interactions.

Subscriber App for Loosely-Coupled Interactions

This section builds a sample custom app named `Loosely Coupled Subscriber`. This sample has two subscriptions:

- One that is an open subscription to accept any published topic payload.
- One that has a specific topic payload to accept location information. This app uses the location information to invoke a mashup and retrieve airport, hotel and limo information for that location.

The tasks involved in building this custom app to support loosely-coupled interactions include:

- HTML and CSS for the Loosely-Coupled Subscriber Sample App
- Declaring Subscribe Topics in the App Specification
- The Initial Subscriber App Constructor
- The Subscription Handler for a Topic with Any as the Payload
- The Subscription Handler for a Topic with Specific Properties
- Wiring Subscription Topics in Mashboard and Testing the Interactions

HTML and CSS for the Loosely-Coupled Subscriber Sample App

We start with the HTML needed for this custom app to receive and render results from subscriptions:

```
<div
>
  <h2>Loosely Coupled Subscriber Sample</h2>
  <table
>
    <thead>
      <tr><th>Airports</th><th>Address</th><th>URL</th><th>Phone</th></tr>
    </thead>
    <tbody
></tbody>
    <thead>
      <tr><th>Hotels</th><th>Address</th><th>URL</th><th>Phone</th></tr>
    </thead>
    <tbody
></tbody>
    <thead>
      <tr><th>Limos</th><th>Address</th><th>URL</th><th>Phone</th></tr>
    </thead>
    <tbody
></tbody>
  </table>
  <h3>Messages Received for Topic Any</h3>
  <div
/>
</div>
```

The `<table>` renders the results of three queries from a mashup. This mashup is invoked by a subscription that has a specific payload with location information. The `<div>` for event messages is used to render results from an open subscription.

The CSS for this example, shown here, is fairly typical. It is *important*, however, that all of the styles are class-based. This pattern ensures that styles for this app will not affect other apps in a workspace or interfere with styles in the various containers where the app or workspace are deployed.

```
div.loose-wiring-sample { font-family: Arial, Helvetica, sans-serif;  }
div.loose-wiring-sample h2, div.loose-wiring-sample h3 { font-weight: bold;
    color: #555; padding-top: 10px; }
div.loose-wiring-sample div, { font-size: 9pt; margin: 5px;}
.event-message { border: 1px solid #CFCFCF; height: auto; overflow: hidden;
    width: 99%; border-bottom: 0px; }
.event-messages { border-bottom: 1px solid #CFCFCF; }
table.stockTable {border: 1px solid #999999; border-collapse: collapse; }
table.stockTable thead {background-color: #ededed; font-size: 9pt;}
```

```
table.stockTable th, table.stockTable td {padding:5px;
   border: 1px solid #999999;}
```

Declaring Subscribe Topics in the App Specification

In addition to the basic changes to the default specification, such as `id`, `name`, and `jsclass`, the App Specification for this example has added the jQuery Templating library in `<requires>`, declared the dependency to a mashup in `<dependson>` and update `<presto-meta>` device compatibility flags.

In order to support loosely-coupled interactions, a `<topics>` section has been added where two subscriptions are declared:

```
<?xml version="1.0" encoding="UTF-8"?>
<app name="Loosely Coupled Subscriber" id="Loosely_Coupled_Subscriber"
jsclass="Sample.LooselyCoupledSubscriber"
    minimizable="false" draggable="false" height="auto" width="600">
  <title>Loosely Coupled Subscriber Sample</title>
  <description>Sample demonstrates an App that subscribes to two topics for loosely-coupled inter
<dependson>
<resource name="TravelInfo_for_Zip" operation="runMashup" type="service"/>
</dependson>
  <properties/>
  <presto-meta name="presto.desktopCompatible" type="text">true</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">false</presto-meta>
<topics>
<topic name="Sample.specificPayload" datatype="object" subscribe="true">
<description>This topic expects location information including a Zip or Postal code. </descriptio
<properties>
<property name="city" datatype="string"/>
<property name="state" datatype="string" />
<property name="country" datatype="string"/>
<property name="postalCode" datatype="string"/>
</properties>
</topic>
<topic name="Sample.anyPayload" datatype="any" subscribe="true">
<description>Use this topic to wire to any published topic. No property wiring is possible becaus
</topic>
</topics>
  <requires>
<require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

The `Sample.anyPayload` subscription is an *open subscription* because the datatype is `any`. With open subscriptions, a custom app accepts published topics with any payload.

The `Sample.specificPayload` has a datatype of `object`, which is the more common pattern for declaring topics. This indicates it accepts messages with a complex content. The expected payload is defined using `<properties` and `<property>` elements.

For more information on declaring topics and message payloads, see "Declare App Topics and Payloads" on page 1285.

The Initial Subscriber App Constructor

The constructor for this example follows common patterns, declaring a namespace for the app, finding the root element for the app and other nodes of interest within the app,

```
Presto.namespace("Sample");
  Sample.LooselyCoupledSubscriber = function( app ) {
  var root = jQuery( app.getRootElement() );
var queryTable = root.find(".queries");
var airportBody = queryTable.find(".airportBody");
var hotelBody = queryTable.find(".hotelBody");
var limoBody = queryTable.find(".limoBody");
var messageDiv = root.find(".event-messages");
var subRowMarkup = "<tr><td>${Title}</td><td>${Address}</td><td>${Url}</td><td>${Phone}</td></tr>
var subRowTemplate = jQuery.template("subRowTemplate", subRowMarkup);
root.find(".queries").hide();
};
```

It also defines a jQuery template that will be used to render and populate rows in the table once a messages for the `Sample.specificPayload` topic is received.

The Subscription Handler for a Topic with Any as the Payload

To receive messages from a subscription, apps must implement the `receive(`*`topic`*`,` *`msg`*`)` method from the MashZone NextGen App API. For more details on this method or other methods for apps, see the MashZone NextGenApp API.

This app has two subscriptions that the `receive` method must handle. For the `Sample.anyPayload` topic, this app receives the message, converts it to JSON and then simply displays the entire message in `<div />` like this:

```
...
  root.find(".queries").hide();
  // App subscription handler
this.receive = function(topic, message) {
message = typeof message == 'object' ? Object.toJSON(message) : message;
messageDiv.append("<div class='event-message'>" + topic + ": " +
message + "</div>");
};
};
```

A more common approach with a open subscription message would be to check for properties of use to the app and use them to update the app. The handler must also handle receiving messages that have no properties of interest. For example:

```
Sample.Subscriber = function(app){
  var showMessage = function(msg){
    //message received logic
  },
showAddress = function(address, city, state){
    //address received logic
  };
  // This App is interested in 2 types of event, one that carries a
  // message property and another which carries properties
  // address, city and state
this.receive = function(topic,msg){
if(typeof msg == 'object'){
if(msg.message){
showMessage(msg.message);
```

```
} else if(msg.address && msg.city && msg.state){
showAddress(msg.address, msg.city, msg.state);
} else {
console.error('unexpected event data received', msg);
}
}
};
}
```

The Subscription Handler for a Topic with Specific Properties

The subscription for the `Sample.specificPayload` topic is the second subscription that this sample must handle in the existing `receive` method. To do this, the handler must be able to differentiate messages received from different topics.

This typically involves checking for specific properties that are unique to each topic payload.

> **Note:** Although both the message and topic name are sent in loosely-coupled interactions, the name that is passed is the publisher's topic name. In tightly-coupled interactions, subscriber apps can test the topic name for a given message because the names are known in advance. With loose-coupling, however, this is not true.

First, we need to add the conditions to determine which subscription topic a given message belongs to. Since we know the `Sample.specificPayload` payload, we can test for those properties:

```
...
  root.find(".queries").hide();
  // App subscription handler
  this.receive = function(topic, message) {
if (message && message.postalCode) {
      //invoke mashup
} else {
message = typeof message == 'object' ? Object.toJSON(message) : message;
messageDiv.append("<div class='event-message'>" + topic + ": " +
message + "</div>");
}
  };
};
```

And finally, we need to handle the `Sample.specificPayload` messages. In this example, we use properties from the message to update input parameters for a mashup that returns three query results for a given location.

The handler invokes the mashup, using the default connection to MashZone NextGen for the app, and two callbacks for successful and failed responses:

```
...
  root.find(".queries").hide();
  // App subscription handler
  this.receive = function(topic, message) {
    if (message && message.postalCode) {
      //invoke mashup
var prestoUrl = "/presto/edge/api/rest/TravelInfo_for_Zip/runMashup?x-presto-resultFormat=json&zi
app.getConnection().request({
url: prestoUrl,
```

```
type: "get",
contentType: "application/x-www-form-urlencoded",
data: ""
},
{ onSuccess: function(response) {
//render table           },
onFailure: function(e) {
app.handleException({
message: 'Failed to find travel information: ' + e.message
});
}
});
    } else {
        message = typeof message == 'object' ? Object.toJSON(message) : message;
       messageDiv.append("<div class='event-message'>" + topic + ": " +
          message + "</div>");
    }
  };
};
```

The `onSuccess` callback must render the results from the mashup using the jQuery template into three separate sets of body rows:

```
...
  root.find(".queries").hide();
  // App subscription handler
  this.receive = function(topic, message) {
    if (message && message.postalCode) {
      //invoke mashup
      var prestoUrl = "/presto/edge/api/rest/TravelInfo_for_Zip/runMashup?x-presto-resultFormat=j
      app.getConnection().request({
          url: prestoUrl,
          type: "get",
          contentType: "application/x-www-form-urlencoded",
          data: ""
        },
        { onSuccess: function(response) {
queryTable.show();
if (response.travelinfo && response.travelinfo.Query){
                  //clear previous body rows
airportBody.empty();
hotelBody.empty();
limoBody.empty();
                  //render body rows for each query from mashup
for (var i=0; i < response.travelinfo.Query.length; i +=1) {
switch (response.travelinfo.Query[i].value) {
case 'airport':
var airports = response.travelinfo.Query[i].Result;
jQuery.tmpl(subRowTemplate, airports).appendTo(airportBody);
case 'hotel':
var hotels = response.travelinfo.Query[i].Result;
jQuery.tmpl(subRowTemplate, hotels).appendTo(hotelBody);
case 'limo':
var limos = response.travelinfo.Query[i].Result;
jQuery.tmpl(subRowTemplate, limos).appendTo(limoBody);
}
};
} else {
queryTable.hide();
}
          },
          onFailure: function(e) {
```

```
                app.handleException({
                    message: 'Failed to find travel information: ' + e.message
                });
            }
        });
    } else {
        message = typeof message == 'object' ? Object.toJSON(message) : message;
       messageDiv.append("<div class='event-message'>" + topic + ": " +
          message + "</div>");
    }
  };
};
```

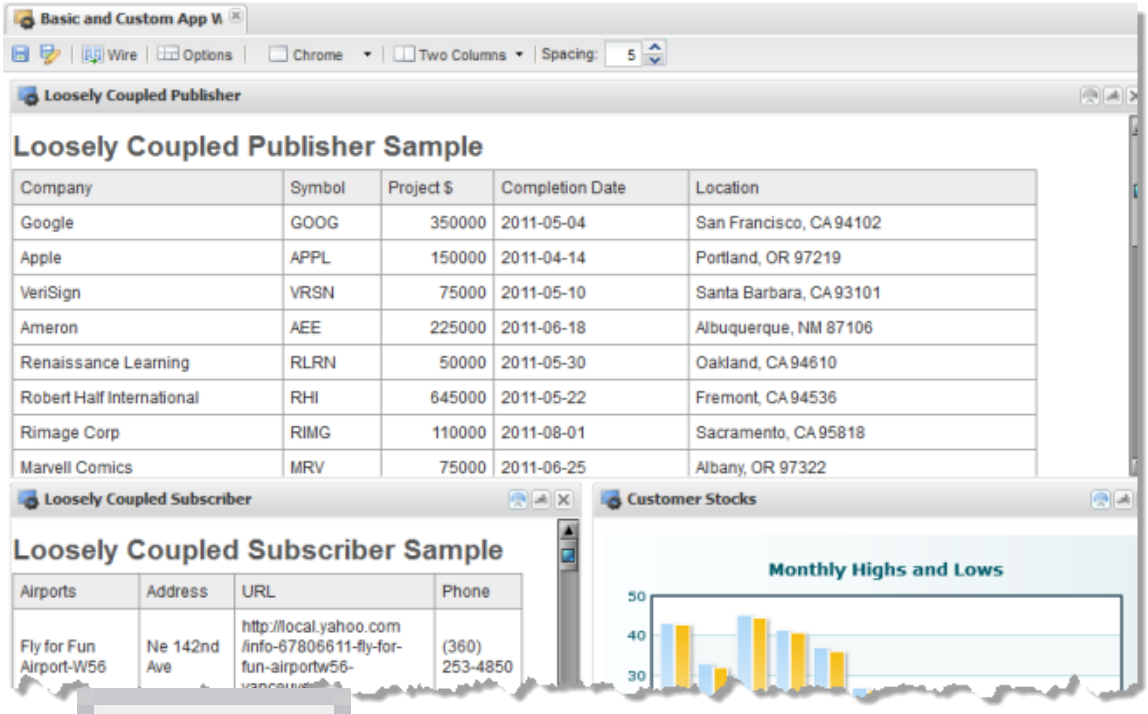
Wiring Subscription Topics in Mashboard and Testing the Interactions

With this sample subscriber app, you cannot easily test it using the app template index.html file as it only renders information when it receives a message for one of it's two subscriptions. To test this app, you must upload it to MashZone NextGen, using the App Editor, add it to a workspace in Mashboard and wire the subscriptions.

First, upload `Loosely Coupled Subscriber` or your custom app using the App Editor (for more information, see "Create Custom Apps from the Base App Package" on page 1226).

To test the interaction

1. Open Mashboard and open the `Basic and Custom Apps Workspace` workspace created in "Publisher App for Loosely-Coupled Interactions" on page 1264. Or start a new workspace and add apps that publish appropriate topics.
2. Open the **Apps** tab and drag `Loosely Coupled Subscriber` into the last cell of the `Basic and Custom Apps Workspace` workspace.

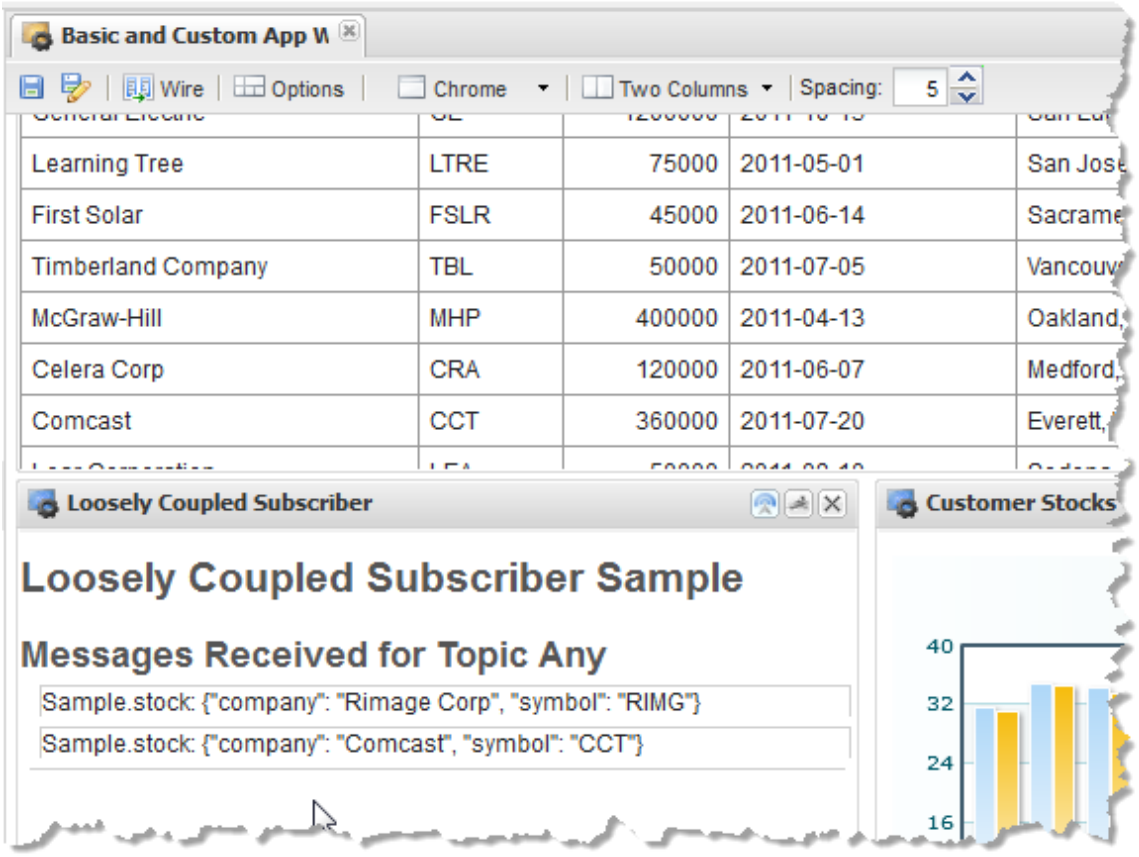


Next, wir `yload` subscription topic:

1. Click **Wire** and choose the `Sample.anyPayload` topic as the subscriber.
2. Choose the `Sample.stock` topic from the `Loosely Coupled Publisher` app as the publisher.
3. Click **Finish**. This saves the wiring configuration and closes the wiring window.

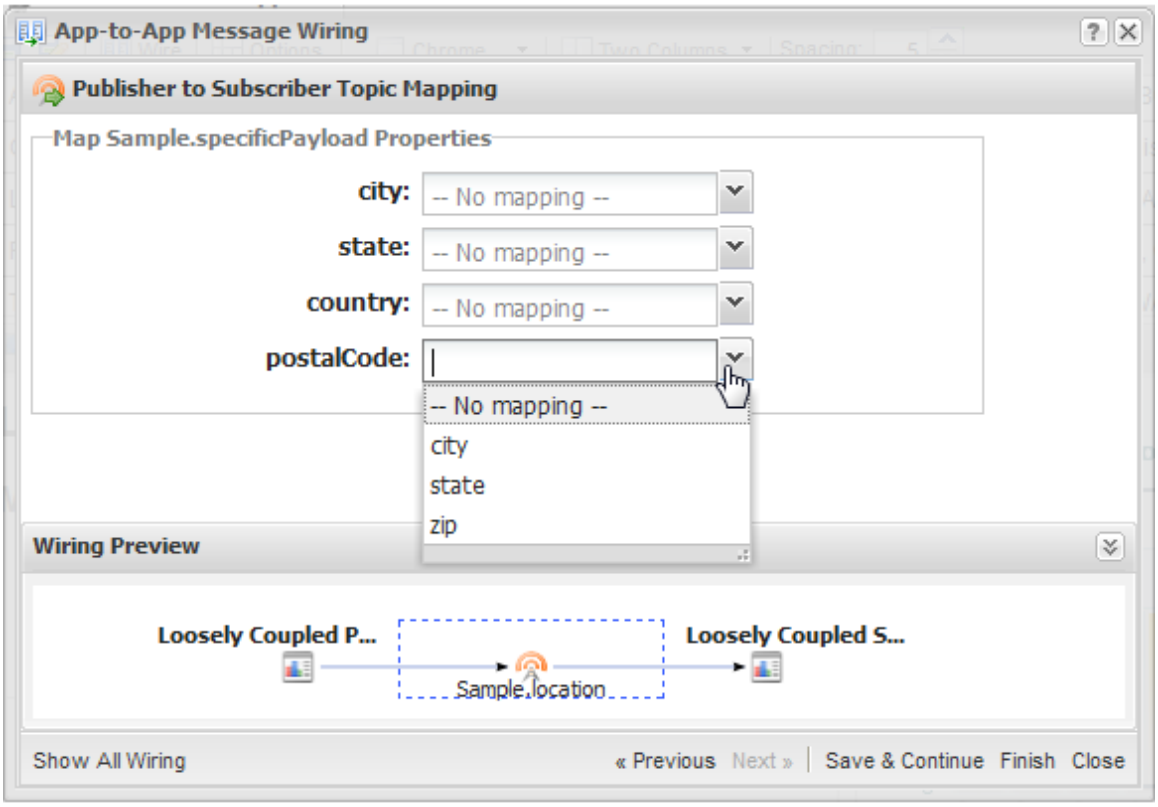
Note: This wiring did not require you to map message payloads. Because the subscription topic is open, with a datatype of `any`, no property mapping is possible.

To test this interaction, click any row in the `Loosely Coupled Publisher` app. This should update the chart in the `Customer Stocks` app and add the raw message data, in JSON format, to the Messages Received section in `Loosely Coupled Subscriber`:



To test the `Sample.specificPayload` subscription in `Loosely Coupled Subscriber`, we must w `Sample.location` publish topic in `Loosely Coupled Publishe`

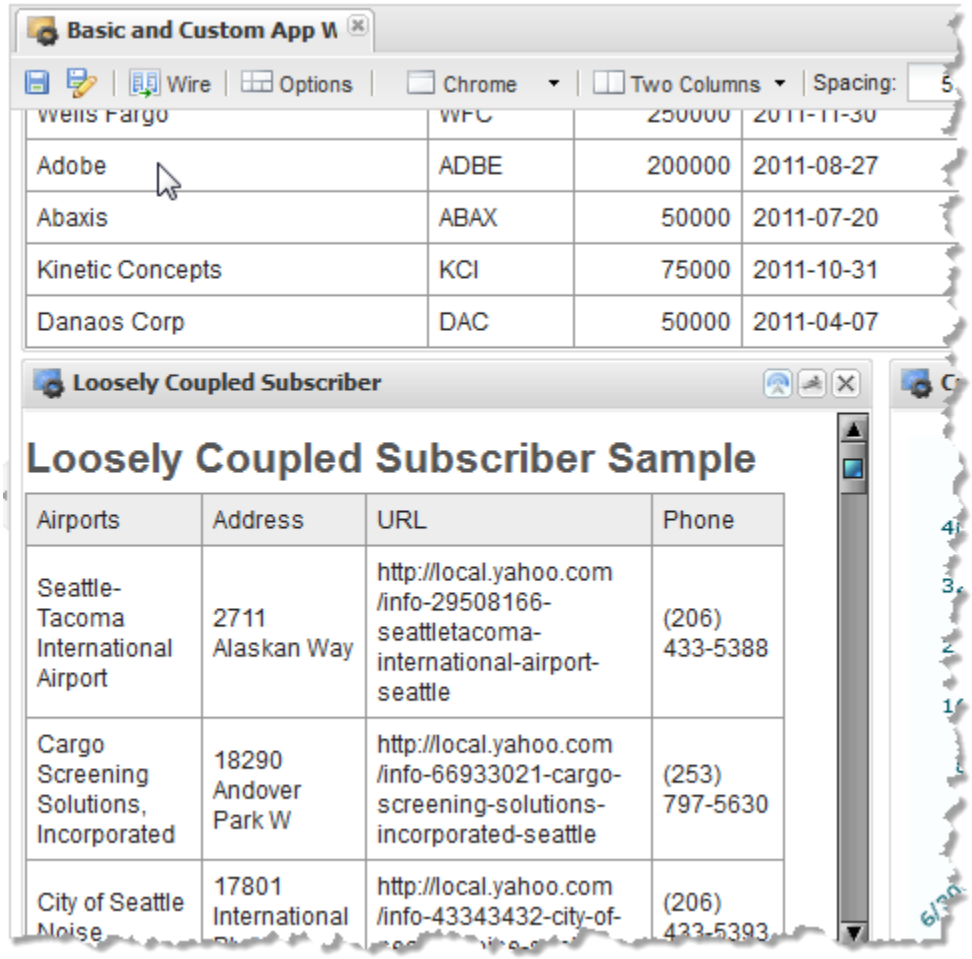
1. Click **Wire** and choose the `Sample.specificPayload` topic as the subscriber.
2. Choose the `Sample.location` topic from the `Loosely Coupled Publisher` app as the publisher.
3. This wiring requires mapping the message payloads, so click **Next**. Map the `zip` property from the published message payload to the `postalCode` property in the expected subscription payload.



4. Click **Finish**. This saves the wiring configuration and closes the wiring window.

To test this final interaction, click any row in the `Loosely Coupled Publisher` app. Three updates should happen:

- This should update the chart in the `Customer Stocks` app.
- It should also add the raw message data, in JSON format, to the Messages Received section in `Loosely Coupled Subscriber`.
- Finally, it should retrieve location queries to the selected client's location and render airport, hotel and limo results in a table in `Loosely Coupled Subscriber`.



Basic App Support for Loosely-Coupled Interactions

Basic apps created in the App Maker wizard can be either publisher or subscriber in a loosely-coupled interaction and thus can interact with custom apps in a workspace. If the app has input parameters, it defines a standard subscription topic named `propertychange` with a payload containing all its input parameters.

Note: *All* input parameters for an app are available for wiring interactions. Settings for the editability and visibility of input parameters in a basic app do not apply.

Basic apps publish topics only if the views selected for that app have events. See "Views for Mashups and Mashables" on page 895 for links to each built-in MashZone NextGen view with information on the events they support.

The Completed Sample Apps

The full JavaScript library for the two sample custom apps used in this topic are shown here:

Completed Library for the Sample Loosely-Coupled Publisher App

```
Presto.namespace("Sample");
Sample.LooselyCoupledPublisher = function( app ) {
    var root = jQuery( app.getRootElement() ),
    projectTable = root.find(".projectTable"),
    projectBody = projectTable.find(".tblBody"),
    rowMarkup = "<tr class='project'>"+
        "<td class='company'>${customer_company}</td>"+
        "<td class='symbol'>${customer_symbol}</td>"+
        "<td class='projectValue' align='right'>${project_value}</td>"+
        "<td class='projectEta'>${project_eta}</td>"+
        "<td>"+
            "<span class='city'>${customer_city}</span>, "+
            "<span class='state'>${customer_state}</span> "+
            "<span class='zip'>${customer_zip}</span>"+
        "</td>"+
    "</tr>",
    rowTemplate = jQuery.template("rowTemplate", rowMarkup),
    //inner function to publish topics for row click
    onRowClick = function(event) {
        var row = jQuery(this),
        customer = row.find("td.company").text(),
        symbol = row.find("td.symbol").text(),
        city = row.find("td span.city").text(),
        state = row.find("td span.state").text(),
        zip = row.find("td span.zip").text();
        app.publish("Sample.stock", {
            "company": customer,
            "symbol": symbol
        });
        app.publish("Sample.location", {
            "city": city,
            "state": state,
            "zip": zip
        });
    },
    //invoke mashup
    prestoUrl = "/presto/edge/api/rest/CustomerProjectSummary/runMashup?x-presto-resultFormat=jsc
    this.onLoad = function(app) {
        app.getConnection().request({
            url: prestoUrl,
            type: "get",
            contentType: "application/x-www-form-urlencoded",
            data: ""
        }, {
            onSuccess: function(response) {
                if (response.result && response.result.customer_project) {
                    var projects = response.result.customer_project;
                    projectBody.empty();
                    jQuery.tmpl(rowTemplate, projects).appendTo(projectBody);
                    //bind handler to cell click event
                    projectBody.find("tr.project").click(onRowClick);
                } else {
                    projectTable.hide();
                }
```

```
            },
            onFailure: function(e) {
                app.handleException({
                    message: 'Failed to retrieve project info: ' + e.message
                });
            }
        });
    };
};
```

Completed Library for the Sample Loosely-Coupled Subscriber App

```
Presto.namespace("Sample");
Sample.LooselyCoupledSubscriber = function( app ) {
  var root = jQuery( app.getRootElement() );
  var queryTable = root.find(".queries");
  var airportBody = queryTable.find(".airportBody");
  var hotelBody = queryTable.find(".hotelBody");
  var limoBody = queryTable.find(".limoBody");
  var messageDiv = root.find(".event-messages");
  var subRowMarkup = "<tr><td>${Title}</td><td>${Address}</td><td>${Url}</td><td>${Phone}</td></t
  var subRowTemplate = jQuery.template("subRowTemplate", subRowMarkup);
  root.find(".queries").hide();
  // App subscription handler
  this.receive = function(topic, message) {
    if (message && message.postalCode) {
      //invoke mashup
      var prestoUrl = "/presto/edge/api/rest/TravelInfo_for_Zip/runMashup?x-presto-resultFormat=j
      app.getConnection().request({
          url: prestoUrl,
          type: "get",
          contentType: "application/x-www-form-urlencoded",
          data: ""
        },
        { onSuccess: function(response) {
              queryTable.show();
              if (response.travelinfo && response.travelinfo.Query){
                  //clear previous body rows
                  airportBody.empty();
                  hotelBody.empty();
                  limoBody.empty();
                  //render body rows for each query from mashup
                  for (var i=0; i < response.travelinfo.Query.length; i +=1) {
                    switch (response.travelinfo.Query[i].value) {
                      case 'airport':
                        var airports = response.travelinfo.Query[i].Result;
                        jQuery.tmpl(subRowTemplate, airports).appendTo(airportBody);
                      case 'hotel':
                        var hotels = response.travelinfo.Query[i].Result;
                        jQuery.tmpl(subRowTemplate, hotels).appendTo(hotelBody);
                      case 'limo':
                        var limos = response.travelinfo.Query[i].Result;
                        jQuery.tmpl(subRowTemplate, limos).appendTo(limoBody);
                    }
                  };
              } else {
                  queryTable.hide();
              }
          },
          onFailure: function(e) {
              app.handleException({
                  message: 'Failed to find travel information: ' + e.message
              });
```

```
          }
      });
    } else {
        message = typeof message == 'object' ? Object.toJSON(message) : message;
       messageDiv.append("<div class='event-message'>" + topic + ": " +
          message + "</div>");
    }
  };
};
```

Declare App Topics and Payloads

You declare the topics that a custom app publishes or subscribes to in the App Specification in the <topics> section. Topics allow apps to participate in loosely-coupled interactions that users can define in Mashboard.

The basic syntax requirements for declaring topics is discussed in "<topics> or <topic>" on page 1316. How you define topic names, datatypes and specify topic payloads, however, impacts how app interactions can be configured.

Topic Name

Topic names *must* be valid JavaScript names and must be unique for a given app.

Tip: It is a best practice to use topic names with two parts (or tokens), in the form *namespace* .*event* . Use the same namespace as your custom app and an event name or type, such as `MyApp.selectProject`.

This helps ensure that topic names are unique and clear to users when they wire them to other apps in Mashboard.

It is better to limit topic names to characters, numbers and the simplest of punctuation or other symbols. Two symbols have a specific meaning for topic names:

- Asterisk (*) cannot be used in a topic name as this character is used as a wildcard character for topic subscriptions.
- Period (.) is used to separate tokens within a topic name. Wildcards can replace tokens to define the set of topics for a subscription.

See "Wildcard Subscriptions for Tightly-Coupled Interactions" on page 1262 for examples.

Simple Topic Payloads

Topics have a datatype that defines what messages for that topic are expected to contain. For simple datatypes, such as `string`, the message is just a single, simple value with no additional properties. For example:]

```
<topics>
  <topic name="sample.number" publish="true" datatype="number"/>
  <topic name="sample.string" subscribe="true" datatype="string"/>
</topics>
```

Complex Topic Payloads: Object

The default datatype for topics is `object` which is used for messages that are complex objects containing several properties. With the `object` datatype, you specify the expected properties using <properties> and <property> within <topic>.

App properties must each be simple types. For topics, however, properties can be complex, containing properties down to any level needed. Properties in message payloads can be arrays, however, you cannot specify that a property in a topic payload is an array in the App Specification.

For example:

```
<topics>
  <topic name="sample.list" publish="true" datatype="object">
    <description>A list of simple properties</description>
    <properties>
      <property name="name" datatype="string"/>
      <property name="age" datatype="number"/>
      <property name="member" datatype="enum" possiblevalues="silver,gold,platinum"/>
      <property name="startdate" datatype="date"/>
      <property name="lastpurchase" datatype="timestamp"/>
    </properties>
  </topic>
  <topic name="sample.complex" subscribe="true" datatype="object">
    <description>A more complex payload.</description>
    <properties>
      <property name="name" datatype="string"/>
      <property name="address" datatype="object">
        <property name="street" datatype="string"/>
        <property name="city" datatype="string"/>
        <property name="state" datatype="string"/>
        <property name="zip" datatype="number"/>
      </property>
      <property name="phone" datatype="number"/>
    </properties>
  </topic>
</topics>
```

The messages that an app publishes may contain properties other than those specified in the topic payload. Any properties not defined in the payload are *not* available for wiring in Mashboard. They are *also not* available to the subscriber app at runtime, unless the subscriber topic uses the `any` datatype.

Open Topic Payloads: Any

Apps can leave message payloads open by using the `any` datatype for a topic. Open topic payloads can be useful when an app is designed to work with a variety of apps that use a common information source, such as a database or a specific system in your organization (CRM, ERP or others). Topics from a common information source typically use a well-known set of data with a common set of names that an open subscription can use reliably.

For example:

```
<topics>
  <topic name="publish.any" publish="true" datatype="any"/>
  <topic name="subscribe.any" subscribe="true" datatype="any"/>
```

```
</topics>
```

Topics with a datatype of `any` for publisher apps can be wired in Mashboard to subscribers at the topic level. If users trigger the event that publishes a message in Mashboard, they can also wire the published message payload to the subscriber payload.

Topics with a datatype of `any` for subscriber apps indicate that the Wiring Manager should pass the published topic on to the subscriber *as-is*, with no changes. You can wire a publisher topic to an open subscriber topic at the topic name level, but no wiring is possible for individual properties.

Because of this, subscription handlers for topics with a data type of `any` must find message properties and handle validation and mapping themselves. For an example, see "Subscriber App for Loosely-Coupled Interactions" on page 1272.

Registering Sample Mashables or Mashups

You can register sample mashable information sources and sample mashups as a starting point for evaluating MashZone NextGen. See "Register Samples Mashables" on page 1287 and "Register Sample Mashups" on page 1288 for instructions.

Register Samples Mashables

Note: The MashZone NextGen samples include a web service for Salesforce.com. This web service *only* works with the default Tomcat port, 8080. If you deploy MashZone NextGen on another application server or use another port, you must register this web service manually.

1. Before you start, you must:
   - Have an Internet connection to successfully register some of the sample mashable information sources.
   - If you use a proxy server in the MashZone NextGen environment, you must also add proxy server configuration to the MashZone NextGen Server before you register samples.
   - If you have changed the password for the default MashZone NextGen`Administrator` account, you must also update the password in the script that registers samples.
2. Open a command window and move to the *MashZoneNG-install* /prestocli folder.
3. Run the appropriate script for your operating system:

Windows	Linux, Mac OS X or UNIX
registersamples.bat	register-samples.sh

Note: This process can take several minutes to complete. You should see confirmation messages for each mashable published by this script.

You may see errors for mashable information sources that cannot be registered. Typically this happens when the provider is temporarily unavailable or timeouts due to network loads.

Register Sample Mashups

For more information on available samples, see "Mashup Samples" on page 659.

To register sample mashups

1. Before you start, you must:
   - Register sample mashable information sources before you register sample mashups. See "Register Samples Mashables" on page 1287 for instructions.
   - If you use a proxy server in the MashZone NextGen environment, you must also add proxy server configuration to the MashZone NextGen Server before you register samples.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter one of the following commands for your operating system, using credentials for a MashZone NextGen administrator account:

Windows	Linux, Mac OS X or UNIX
If you installed MashZone NextGen on your local computer using the default TomEE ports, enter: publish-mashups.bat -u *admin-user-name* -p *admin-password*	If you installed MashZone NextGen on your local computer using the default TomEE ports, enter: publish-mashups.sh -u *admin-user-name* -p *admin-password*
If the MashZone NextGen Server is not located on your computer or you used a different port number when you installed MashZone NextGen, enter: publish-mashups.bat -u *admin-user-name* -p *admin-password* -url http://*app-server:port* /presto/edge/api Using the correct domain or IP address and port for the MashZone NextGen Server.	If the MashZone NextGen Server is not located on your computer or you used a different port number when you installed MashZone NextGen, enter: publish-mashups.sh -u *admin-user-name* -p *admin-password* -url http://*app-server:port* /presto/edge/api Using the correct domain or IP address and port for the MashZone NextGen Server.

If the default MashZone NextGen administrator account is unchanged, you can use `Administrator` as the user name and `manage` as the password.

Override Browser Caches for Updates to App Resources

When you update the resource files for customized basic apps or fully custom apps, testing in the App Editor shows the update. However, users may still report problems or see old content because browsers use cached versions of the resource until the cache expires or users manually clear their browser cache. Having users manually clear their browser cache is generally not an acceptable solution.

You can manage browser caching to ensure that they retrieve updated resources for customized or fully custom apps by using the `version` attribute in the "<require>" on page 1314 statement in the App Specification.

Note: This attribute has two different meanings for resources depending on whether the resource is a named library bundled in MashZone NextGen or it is a resource upload within a custom app. See the "<require>" on page 1314 topic for more information.

The MashZone NextGen App framework uses this metadata to append a unique string to the URL used to retrieve that resource for a custom app. Setting this attribute or updating it each time you update a resource ensures that the URL is unique so that browsers always retrieve the updated file.

For this initial example:

```
<app id="SampleTable" name="SampleTable"
   jsclass="Sample.HelloWorld" height="200" width="200"
   draggable="false" minimizable="false">
  <title>SampleTable</title>
  ...
  <requires>
    <require src="js/app.js" type="script" version="1"/>
    <require src="css/app.css" type="css" version="1"/>
    <require src="html/app.html" type="html" version="1"/>
  </requires>
</app>
```

If you update the JavaScript class for this custom app, you simply also change the App Spec to this:

```
<app id="SampleTable" name="SampleTable"
   jsclass="Sample.HelloWorld" height="200" width="200"
   draggable="false" minimizable="false">
  <title>SampleTable</title>
  ...
  <requires>
    <require src="js/app.js" type="script" version="2"/>
    <require src="css/app.css" type="css" version="1"/>
    <require src="html/app.html" type="html" version="1"/>
  </requires>
</app>
```

To ensure your changes are visible to all appropriate users in MashZone NextGen Hub. You may also need to publish the app to the AppDepot to make your changes visible to users in the AppDepot or the MashZone NextGen Mobile apps.

The Structure of a MashZone NextGen App

When apps are loaded, rendered and used in published destinations, this also loads the MashZone NextGen App API and wraps the app in a framework based on the type of app:

- App Framework for Workspace and Custom Apps
- App Framework for Basic Apps

App Framework for Workspace and Custom Apps

The App Framework for workspace apps or custom apps is shown below:

© 2013 Software AG. All rights reserved

In desktop browsers, the app is loaded and rendered in a *container page* in that destination. This page has a *container element* that defines where the app is rendered.

In mobile devices, the app has no container, but instead is rendered in the full screen of the device.

The App Framework wraps the app within the container element or device. It contains the following elements:

- <*iframe*>: in desktop browsers, the app can be rendered *inline* within the container element or *secured* within an <iframe> to isolate the JavaScript and CSS for the app from the container page.
- *Root Element*: the root node that entirely wraps the app itself and all elements supplied by the App Framework. Apps can get a reference to this node using the `getRootElement` method in the MashZone NextGen App API.

  The root node also has a unique class `app-`*`app-name`* that contains the name for this app. This class can be used in CSS selectors to override or add to styles for components within the app framework or the app content.
- *App Title Bar*: by default this element displays the app title and a set of toolbar buttons:
  - **Tools**: opens the input form in both desktop and mobile devices to allow users to change input parameters for the app. Only present if the app has input parameters. Basic apps may have additional forms.
  - **Refresh**: in desktop browsers, allows users to manually refresh the app. Always present for basic apps. See "Enable User-Initiated or Automatic Refreshes" on page 1243 for instructions on enabling this button in custom apps.
  - **Open in New Window**: in desktop browsers, allows the app to open in a separate window or tab outside of the container.
  - **Collaboration Menu**: in desktop browsers, contains menu options to share the app, get embed code for the app, publish the app to SharePoint or edit user preferences for the app, if appropriate.

  In desktop browsers, the container where an app is published can hide this default title bar completely or override the default title bar with its own title bar. Apps can also hide this title bar.

  Note: Hiding the title bar also hides the standard toolbar buttons. This prevents users from updating input parameters for apps or handling other functions such as refreshing data.
- *App Messages*: a default element to display exceptions or error messages using the MashZone NextGen App API. See "Handle Exceptions" on page 1247 for more information and examples.
- *App Content*: the parent within the App Framework where the app is rendered. Apps can listen to this wrapper for resize events or use the App API to handle size changes. See "App Dimensions and Resizing" on page 1236 for more information and examples.

App Framework for Basic Apps

The App Framework for basic apps has the same basic layers presented earlier for workspace and custom apps, as shown in the following figure. In desktop browsers,

there is a container page and within that page a container element where the app is rendered. In mobile devices, the app is rendered in the full screen of the device.

The App Framework wraps the app in a root element that contains the app title bar and an area for messages.

© 2014 Software AG. All rights reserved

It has a container for the app content and then a wrapper for the basic app. This contains:

- *Basic App Tools*: this may include forms for input parameters, sorting and filtering based on configuration for the basic app. This opens from the Tools button in the app title bar.

- *Messages*: an area for the text of the standard "loading data" message and another for other messages.
- *Views Container*: wrapping all views in the basic app.
- *View 1-n* : a wrapper for each view. This contains areas for both the:
  - *View Title Bar* which may be disabled based on user configuration. It is also disabled by default when basic apps are included in workspaces, although users can override that.
  - *View Content* for the grid, chart or map for that view.

  The view wrapper also has a unique class `view-view-name` that contains the name for this view. This class can be used in CSS selectors to override or add to styles for the view and view title bar.

Parameters for App URLs

You can include any of the standard MashZone NextGen header/parameters in URLs to run and display apps. (Standard headers/parameters can also be used in URLs to invoke mashables or mashups.) See "MashZone NextGen Headers/Parameters" on page 1511 for a complete list.

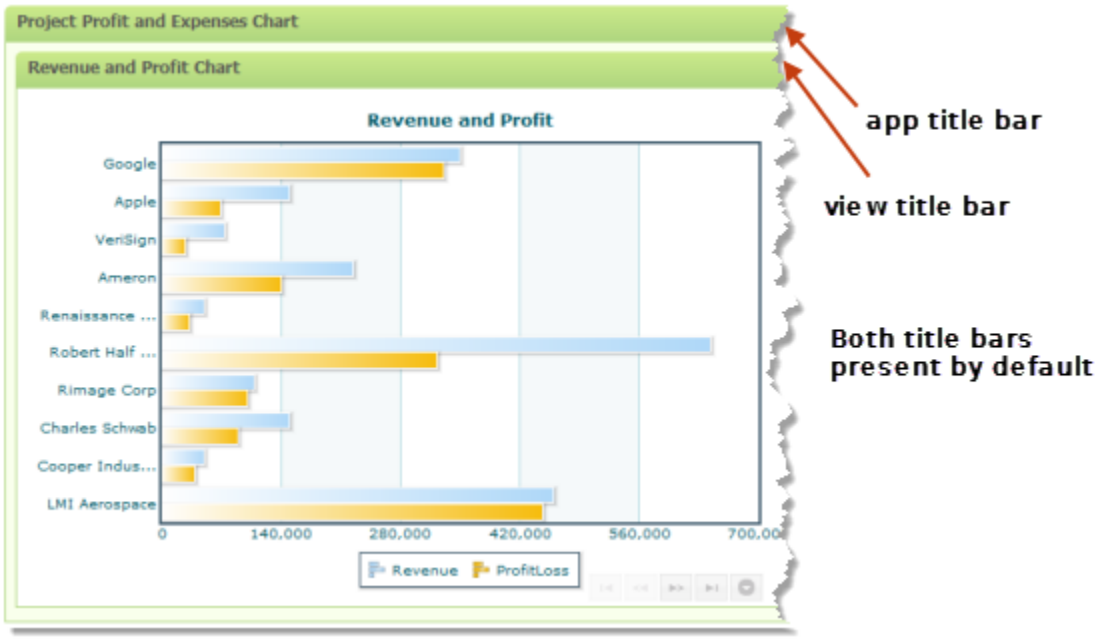
In addition to the standard MashZone NextGen parameters, you can use any of the following parameters in URLs for apps to control specific visual aspects or behavior for apps:

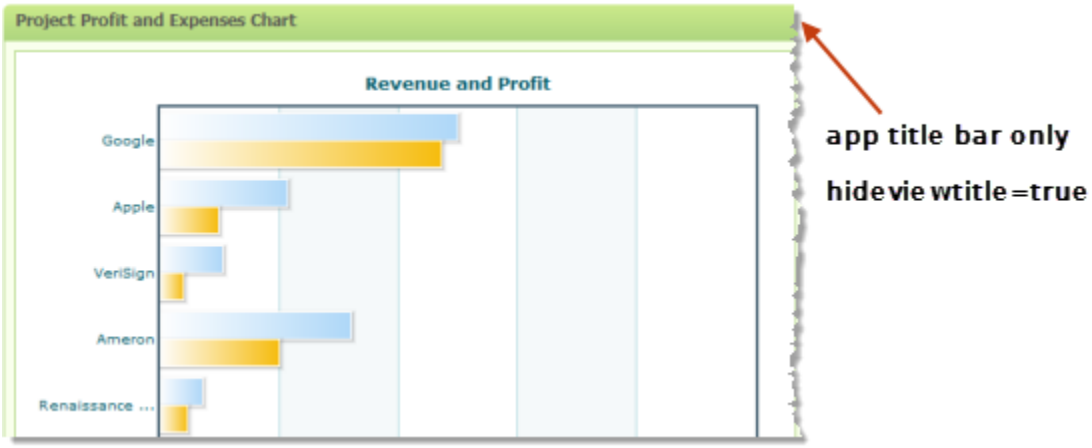
autorefresh	Turns on autorefresh for this app to invoke the underlying mashable or mashup at a frequency defined by the `refreshinterval` parameter. If `refreshinterval` is omitted, this parameter has no affect. For example: `http://localhost:8080/presto/hub/applauncher.html#mid=Project_Management_Summary?autorefresh=true&refreshinterval=300`
debug	By default, MashZone NextGen apps disable the browser console which is commonly used by developers during debugging to show messages logged to the console. Add the `debug` parameter to enable the browser console. No value is required with this parameter. For example: `http://localhost:8080/prestohub?debug` Note that `debug`*must* occur in the app's URL before the `#` sign. For example:

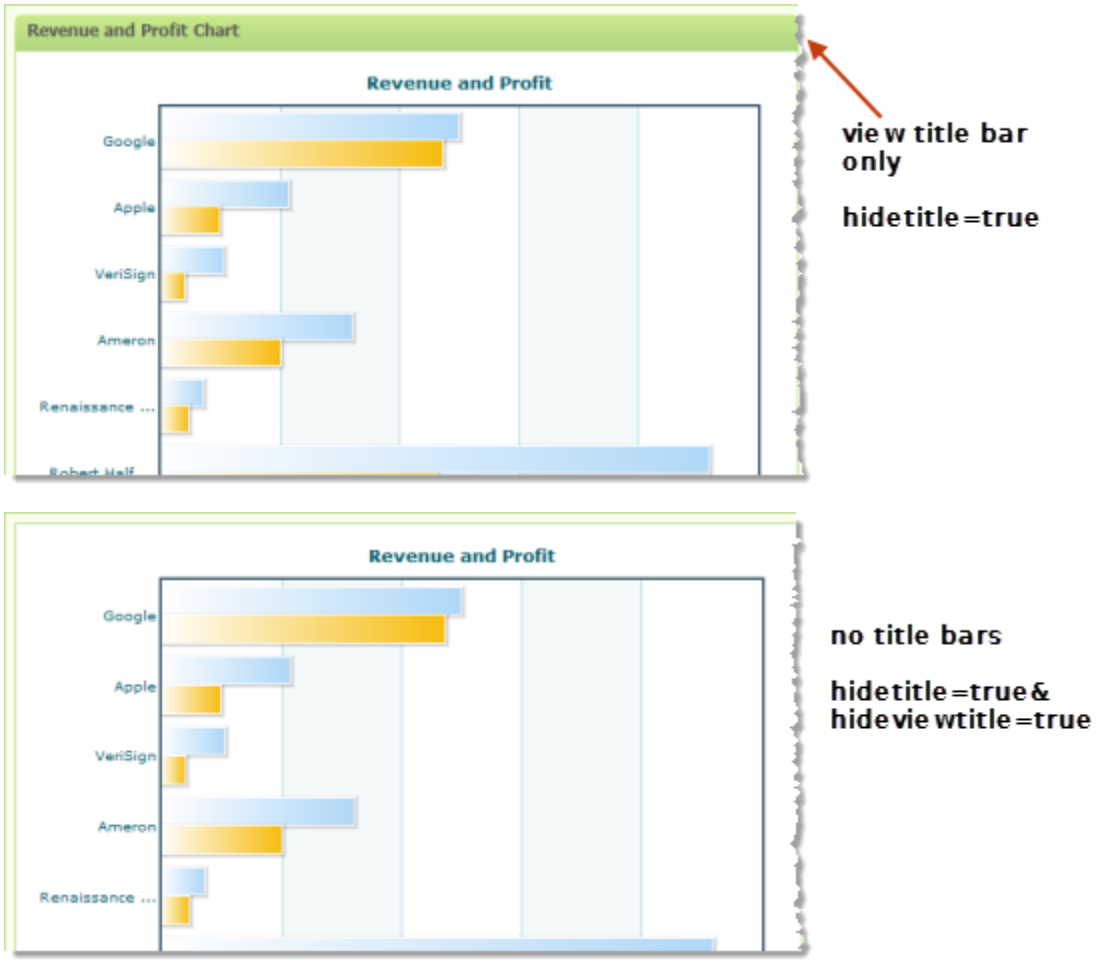
	`http://localhost:8080/presto/hub/search.html?debug#subType=RSS` If the URL includes a redirect, the URL of the final target page must also include the `debug` parameter or the console will remain disabled.
el	The ID of the element in the container page where this app should be rendered. If this parameter is omitted, the app is rendered as a sibling following the <script> element that embeds the app in the container page. For example: `http://localhost:8080/presto/hub?mid=Project_Profit_and_Expenses_Chart&el=profit`
height	The preferred height in pixels for this app. This is not determinate as the container has final control over sizing.
hideinputs	Suppresses the initial opening of the input parameter form for an app. The most typical usage is to hide input parameter forms when they have been wired in a workspace to a published event from some other app in the workspace. For example: `http://localhost:8080/presto/hub/applauncher.html#mid=Project_Management_Summary?hideinputs=true` Omitting this parameter is identical to using a false value.
hidetitle	Suppresses the app title bar in desktop browsers. In mobile devices, the app title bar disappears automatically regardless of this parameter, but is still accessible. For example: `http://localhost:8080/presto/hub/applauncher.html?mid=Project_Profit_and_Expenses_Chart&hidetitle` Suppressing the app title bar also makes tools, such as the form for input parameters, sorting and filtering, unavailable to app users. See also the `hideviewtitle` parameter and "Effects of Title Bar Parameters" on page 1297 for examples of the effects of these parameters.
hideviewtitle	Suppresses view title bars in apps with multiple views in desktop browsers. This parameter has no affect on view title bars in mobile devices.

	For example: `http://localhost:8080/presto/hub/applauncher.html?mid=Project_Profit_and_Expenses_Chart&hideviewtitle` See also the `hideviewtitle` parameter and "Effects of Title Bar Parameters" on page 1297 for examples of the effects of these parameters.
locale	Reserved for future use.
refreshinterval	The number of seconds between automatic invocations of the underlying mashable or mashup to automatically refresh app data. If the refresh interval is set, the autorefresh parameter is automatically true. For example: `http://localhost:8080/presto/hub/applauncher.html#mid=Project_Management_Summary?refreshinterval=300`
sandbox	Forces the app to be rendered inside an `<iframe>` element to ensure that JavaScript and CSS for the app does not conflict with JavaScript and CSS in the containing page. However, the `<iframe>` wrapper also disrupts interactions with other apps in the page (both tightly coupled and loosely coupled interactions). Setting the `sandbox` parameter ensures that published messages for events from user interactions are supported across the `<iframe>`. This is equivalent to the secured property for workspaces. `http://localhost:8080/presto/hub/applauncher.html#mid=Project_Management_Summary?sandbox=true`
singleton	Reserved for future use.
width	The preferred width in pixels for this app. This is not determinate as the container has final control over sizing.

Effects of Title Bar Parameters







App Specification Reference

App specifications define basic information or *meta-data* about an app as well as other resources (files) and properties for the app. App specs also define the topics that the app either publishes or subscribes to support inter-app communication.

The XML elements you may use in an app specification include:

Purpose	Elements
Root	<app>
Metadata	<authors> or <author>
	<dependson>

Purpose	Elements
	<description>
	<icons> or <icon>
	<presto-meta>
	<title>
Resources	<requires> or <require>
	<help>
Properties and datatypes	<properties> or <property>
Inter-App Communication	<topics> or <topic>

Element names in the form `mashzone-`*name* are reserved for MashZone NextGen internal use.

For links to topics with App specification examples, click **More...** above.

<app>

The root element of an app specification.

Can Contain	<title>?, <description>?, <dependson>?, <help>?, (<authors> | <author> | <icons> | <icon> | <presto-meta> | <properties> | <property> | <content> | <requires> | <require> | <topics> | <topic>)*

Attributes

Name	Required	Description
abouturi		A URL to information provided by the developer about this app.
autorefresh		Determines whether a basic or custom app should automatically refresh data (`true`) at a

Name	Required	Description
		specific interval. This defaults to `false`. See also `refreshinterval`. Custom apps must also implement a handler method for refreshes. See Enable User-Initiated or Automatic Refreshes for more information.
debug		Determines whether logged messages display in the console of browser developer tools when the app is run. This defaults to `false`. Developers may set this flag to `false` in the App Specification or use this as an app parameter in the app URL.
height		The default height, in pixels, to render this app or `auto`. Defaults to `auto`. This attribute is only used when the containing page for the app does not specify or control the dimension of the space where the app is rendered. For more information, see App Dimensions and Resizing.
hideheader		Determines whether the title bar displays for this app (`false`). Set to `true` to hide the title bar. The title bar contains the app title and may also contain toolbar buttons based on configuration for the app. See Title Bars and Toolbar Buttons in Apps for more information.
hideinputs		Determines whether the input form, if any, for this app opens *initially* when the app opens (`false`). The input form may still be accessible to users in a toolbar button in the app title bar or through other user interface widgets provided by the container where the app is published. This attribute defaults to `false`. **Note:** This attribute has no effect for basic apps that have configured the input form as a view.
id		Unique identifier assigned by MashZone NextGen when the app is created. This is read-only and cannot be changed.

Name	Required	Description
name	yes	The name for this app. MashZone NextGen uses the app name to assign a unique identifier to the app. App names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`
jsclass		The constructor function for the optional JavaScript class for this app. Apps are not required to define a Javascript class with a constructor *unless* the app should be initialized after the page is ready. Initializing the app after the page is ready can ensure that specific properties or objects are available.
refreshinterval		How frequently, in seconds, a basic or custom app should automatically refresh data. This defaults to zero. If set to a number greater than zero, `autorefresh` is treated as true. Custom apps must also implement a handler method for refreshes. See Enable User-Initiated or Automatic Refreshes for more information.
sandbox		Determines whether this app should be rendered in a secure mode (`true`). This is `false` by default. **Note:** This property only applies to apps when they are used in Mashboard, SharePoint or embedded in HTML pages, wikis or other environments. In secure mode, the app runs inside an <iframe> to isolate the app from effects from other objects in the containing page and vice versa.
singleton		This is reserved for future use.
version		The version of this app. This is informational only.
width		The default width, in pixels, to render this app or `auto`. Defaults to `auto`.

Name	Required	Description
		This attribute is only used when the containing page for the app does not specify or control the dimension of the space where the app is rendered. For more information, see App Dimensions and Resizing.

<authors> or <author>

<authors>

A list of authors for this app.

Can Contain	"<author>" on page 1302*
Allowed In	<app>

<author>

An author for this app.

Can Contain	Text with an author's name or empty when attributes provide additional information.
Allowed In	<app> or <authors>

Attributes

Name	Required	Description
email		Optional author's email address.
location		Optional location, such as a city or country, for this author.
name	Conditional	Required if the author's name is not provided as text within the <author> element.
organization		Optional company or organization affiliation for this author.

<dependson>

Lists all the mashups, mashable information sources or other apps that this app uses. Dependency information is not required for custom apps, but is *highly* recommended. MashZone NextGen uses dependency information to help ensure artifacts are managed properly.

Can Contain	"<resource>" on page 1303+
Allowed In	<app>

<resource>

One mashable information source, mashup or app that this app is dependent on. Only one level of dependency information is needed for those artifacts that the app calls directly.

Attributes

Name	Required	Description
name	yes	The MashZone NextGen ID for the mashable, mashup or app for this dependency.
operation		The ID for the specific mashable operation that the app is dependent on.
type		The type of this artifact in MashZone NextGen. This defaults to `service` for mashups or mashable information sources. It can also be `app` for workspace dependencies on apps.

<description>

A description of this app, property or topic.

Can Contain	Text or well-formed HTML.
Allowed In	<app>, <property>, or <topic>

<help>

An HTML page to use as the help topic for this app. This HTML file can be hosted in:

- Another web server or application server.
- The app package for this custom app.
- The MashZone NextGen Server as a separate resource.

Images, CSS stylesheets or other resources used in the help topic should be hosted with the help topic.

Custom app can make this help available to users through a button or other UI widget using methods from the MashZone NextGenApp API.

Can Contain	Empty
Allowed In	<app>

Attributes

Name	Required	Description

href	yes	The URL to the HTML help topic for this app. This URL can be:
		■ Absolute, for an HTML page that is hosted in a web server or application server that is accessible to the MashZone NextGen Server. For example:
		`http://myOrg.com/apps/help/MyAppHelp.html`
		■ Relative to the root folder for the app. For help topics that are uploaded within the app package, the URL must *not* start with a slash. For example:
		`html/help.html`
		■ Relative to the MashZone NextGen Server, if a MashZone NextGen administrator has uploaded the HTML page separately to the MashZone NextGen Server.
		For help topics that are hosted in the MashZone NextGen Server independently of their app, the URL uses the form `/presto/files/`*path/assigned-file-name* based on the path and

Name	Required	Description
		file name assigned when the help topic was uploaded. For example: `/presto/files/help/MyAppHelp.html`
target		A target name of the window where this help topic should open. By default, help topics open in a new browser window with a target of `_blank`.
title		An optional title to use for the help window where this topic opens. This defaults to `Help`.

<icons> or <icon>

<icons>

This element is *reserved* for future use.

Can Contain	"<icon>" on page 1305*
Allowed In	<app>

<icon>

This element is *reserved* for future use.

Can Contain	Empty.
Allowed In	<app>

Attributes

Name	Required	Description
src	yes	The URL to this image.
height		The height, in pixels, of this image.
width		The width, in pixels, of this image.

<presto-meta>

This element is reserved for internal use by MashZone NextGen with these exceptions:

- App Layout
- App Themes
- Charts Theme
- Device Compatibility Flags
- Toolbar Button Flags
- View Configuration

Important: You should *never* alter other <presto-meta> elements generated by MashZone NextGen for basic apps or workspace apps.

App Layout

The type of layout for basic apps is defined in `<presto-meta name="layout">`. The content identifies the selected layout:

`combo`

Each view appears in its own card. Users choose the current card from a pull-down list.

`stacked`

Views are stacked vertically, one after another.

`tab`

This is the default layout. Each view appears in its own tab.

`two-column`

Views are tiled across two columns per row, with enough rows to render all views.

`wide-top-row`

Similar to the two column layout except the very first row contains a single view that is rendered across both columns.

`custom`

The layout for the views is user defined in an HTML file, plus optional CSS file, that have been added to the basic app. See "Customize the Layout for Basic Apps" on page 1207 for instructions.

App Themes

Basic apps have a property, `Theme`, that defines the built-in theme which is applied to borders, toolbars and backgrounds for the App Framework:

See "Themes" on page 1107 for an example of the areas affected when basic apps are published individually. When basic apps are included in workspace apps, the workspace overrides the themes of individual apps within the workspace.

You can change the theme assigned to a basic app using any of the valid values for this property:

You can also customize themes. See "Customize Themes for Basic Apps and Views" on page 1200 for more information.

See also **App styling** in "Edit App Properties/Themes" on page 1107.

Charts Theme

You can alter <presto-meta> elements with a `chartstheme` name to change the assigned charts theme. The charts theme provides a global layout configuration for all charts that are included in a basic app.

In the following example the `ocean` charts theme ist assigned:

```
...
<presto-meta name="chartstheme" type="text">ocean</presto-meta>
...
```

By defaultMashZone NextGenprovides the **carbon**, **fint**, **ocean** and **zune** charts themes.

See also **Chart styling (default themes and custom themes)** in "Edit App Properties/Themes" on page 1107.

Device Compatibility Flags

You can alter the following <presto-meta> elements that define device-compatibility for an app:

- `<presto-meta name="presto.desktopCompatible" type="text">`
- `<presto-meta name="presto.phoneCompatible" type="text">`
- `<presto-meta name="presto.tabletCompatible" type="text">`

The value for these flags can be `true` or `false`. For example:

```
<app id="MobileTabletTable" name="MobileTable"
   jsclass="MyOrg.MobileTabletTable" height="200" width="200"
   draggable="false" minimizable="false">
  <title>Tablet Table</title>
  ...
```

```
  <presto-meta name="presto.desktopCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.phoneCompatible" type="text">false</presto-meta>
  <presto-meta name="presto.tabletCompatible" type="text">true</presto-meta>
  ...
  <requires>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

Toolbar Button Flags

You can alter the following <presto-meta> elements that define visibility for two buttons in the app title bar:

- `<presto-meta name="presto.allowOpenNewWindow" type="text">`: this controls visibility for the **Open in New Window** toolbar button for the app.

  See "Title Bars and Toolbar Buttons in Apps" on page 1112 for more information.

- `<presto-meta name="presto.shareable" type="text">`: this controls the visibility for the **Share Menu** toolbar button for the app.

  See "Title Bars and Toolbar Buttons in Apps" on page 1112 for more information

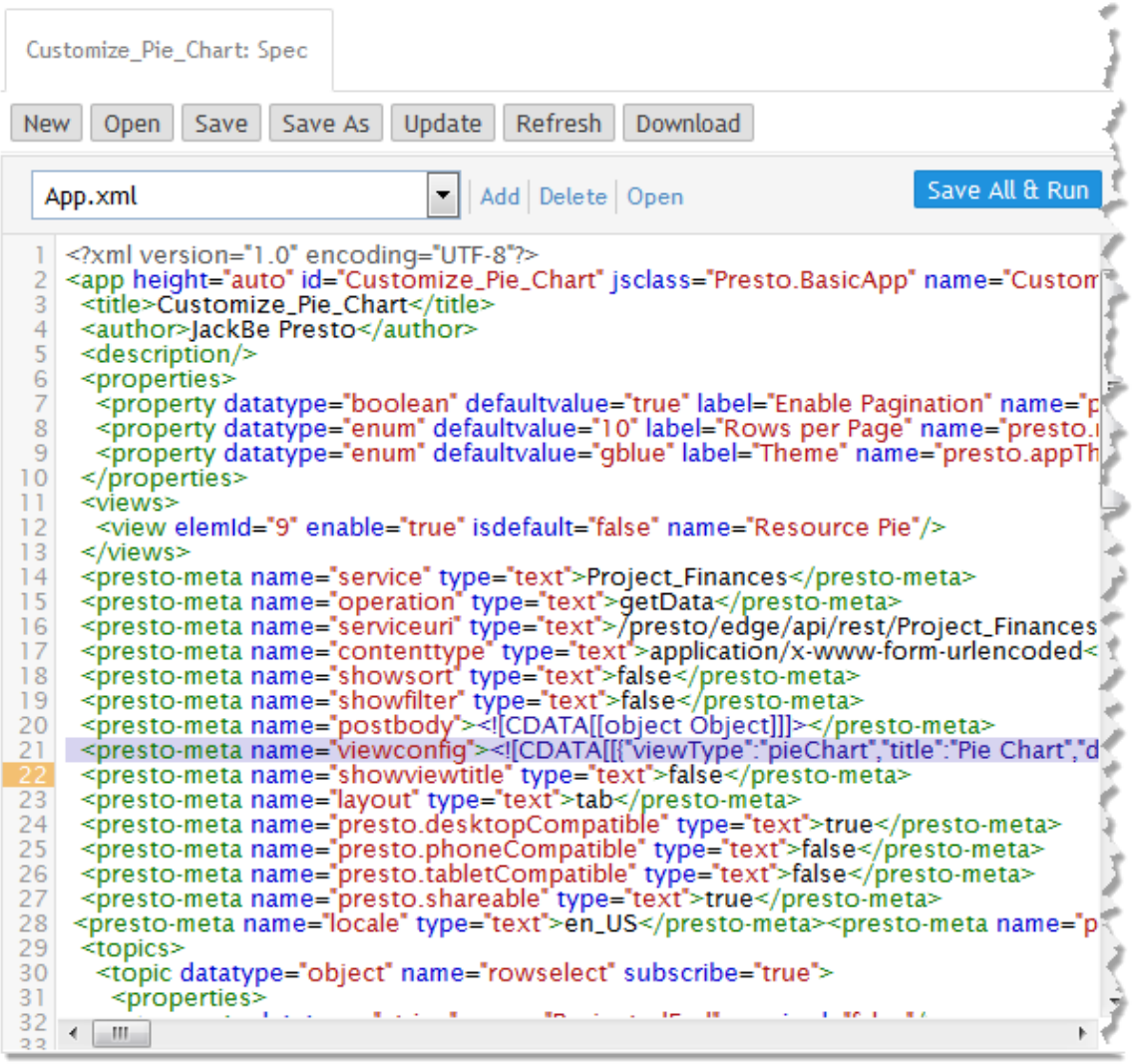
The value for these flags can be `true` or `false`. For example:

```
<app id="NotShareable" name="Not Shareable App"
   jsclass="MyOrg.NotShareable" height="200" width="200"
   draggable="false" minimizable="false">
  <title>Not Shareable App</title>
  ...
  <presto-meta name="presto.shareable" type="text">false</presto-meta>
  ...
  <requires>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
  </requires>
</app>
```

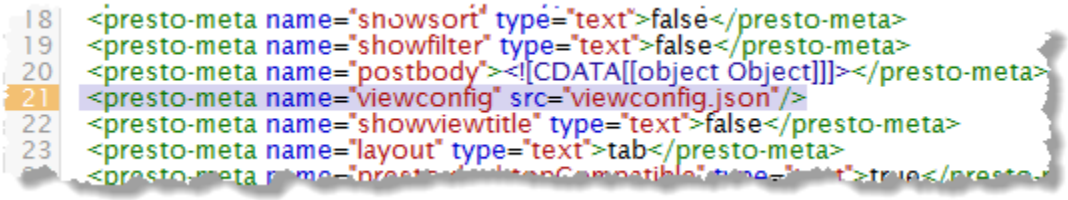
View Configuration

You can alter <presto-meta> elements with a `viewconfig` name to point to a separate JSON configuration file to provide view configuration for all views that are included in a basic app.

For basic apps, this meta-data element initially contains a CDATA marked section that contains an array with JSON objects, each of which is configuration for one view, such as this example:



You can move this configuration to a separate file to make it easier to customize this configuration and then update this `<presto-meta>` element to point to that separate file, such as this example:



See "Customize Built-In Chart Attributes" on page 1218 for instructions and an example of this process.

<properties> or <property>

<properties>

A list of properties for this app. Properties include input parameters for the app, options to determine look and feel, such as width, or options to control other behavior. For more information and examples of declaring properties, setting property values or retrieving properties or property values, see "Declare, Get and Set Properties in Custom Apps" on page 1227.

Properties are *also* used to define the *payload* or data that the app may publish to other apps or receive from other apps. The payload is defined in <topic> elements. Properties for topic payloads may be complex, containing other properties in any structure needed. See "Declare App Topics and Payloads" on page 1285 for more information and examples.

Can Contain	"<property>" on page 1310 +
Allowed In	<app>, <topic> or <property> (only within <topic>)

<property>

A property of an app or of a complex object for a topic payload.

Can Contain	(<property> | <properties>)*
Allowed In	<app>, <properties> or <property> (only within <topic>)

Attributes

Name	Required	Description
datatype	yes	The datatype for this property. This defaults to `string`. Valid datatypes include: ■ `string` ■ `number` ■ `boolean` ■ `date` ■ `timestamp` = currently only available within topic payloads

Name	Required	Description
		■ `enum` = an enumerated list of valid values. You set the values that users can select from in the `possiblevalues` attribute. ■ `any` = any valid datatype is accepted for this property. ■ `object` = a complex property only allowed within topic payloads. This is the default type for topic payloads.
defaultvalue		An optional default value to use for this property if it is not otherwise specified.
isinput		Determines whether this property is treated as an input parameter for the app (`true`). This defaults to `false`. This attribute is used in combination with the `visibility` attributed to determine access to the property in different contexts. See "Controlling Property Access and Visibility" on page 1312 for details.
label		An optional label to display to users for this property. **Note:** MashZone NextGen displays labels for properties in the input parameter forms that are generated for basic apps and also when users edit app properties in MashZone NextGen.
name	yes	The name of this property.
possiblevalues		A short list of a valid values, separated by commas, for this property. This only applies if `datatype = 'enum'`. **Note:** MashZone NextGen uses and enforces valid value lists for properties only in the input parameter forms that are generated for basic apps or when users edit app properties in MashZone NextGen.

Name	Required	Description
required		Determines whether this property must be supplied `true` or not for the app to run successfully. This defaults to `false`. **Note:** MashZone NextGen enforces required values for properties only in the input parameter forms that are generated for basic apps.
tooltip		A short, optional description to use as a tooltip or hint to users for this property. **Note:** MashZone NextGen displays tooltips for properties in the input parameter forms that are generated for basic apps.
validation		For string properties, the basic validation rules to apply when users enter a value. Valid validation rules include: - `any` = no validation is applied. This is the default. - `alphanumeric` = must be an alphanumeric character. - `alpha` = must be an alphabetic character. - `numeric` = must be a number. **Note:** MashZone NextGen enforces validation only in the input parameter forms that are generated for basic apps.
visibility		Can be `visible`, `read-only` or `hidden`. This determines visibility and user access to a property, in combination with the `isinput` attribute in several different contexts. See "Controlling Property Access and Visibility" on page 1312 for details.

Controlling Property Access and Visibility

Whether a property is considered an input parameter for the app and its visibility affects whether the property can appear in the input form for the app. It also affects whether it can be used in wiring interactions between apps in workspaces created in Mashboard and whether the property can be saved as a preference for user favorites in the AppDepot.

Note: These attributes do not control programmatic access to app properties.

To enable the use of a property in:

- *App Input Forms*: set `isinput = true` and `visibility = visible`.

  With input forms, users can change the property whenever they want to. However, users cannot save their preferences with a new favorite copy of an app in AppDepot.

- *Wired Workspace Interactions*: set `isinput = true` and `visibility = visible`.

  Note: In earlier releases, this use required that the property have its visiblity hidden. This is no longer required. Instead, you can hide input forms for apps in workspaces in Mashboard. See "Hiding Input Forms for Apps in a Workspace" on page 1181 for instructions.

- *Saved Preferences for Favorites in AppDepot*: set `isinput = true` and `visibility = visible`.

  Users can update a property when they mark an app as a favorite in AppDepot and save that update as a 'personal copy' with a different name. When they use this favorite, the saved preferences are automatically applied.

  Note: In earlier releases, this use required that the property have its visiblity hidden. This is no longer required.

- *Use Default Only, Either Hidden or Read-Only*: to omit the property from input forms, wiring, or any customization, set `isinput = false`. Then use `visibility` to make the property completely hidden or read-only

<content>

This element is reserved for internal use by MashZone NextGen for workspace apps only. It is maintained by MashZone NextGen and should *not* be altered.

<requires> or <require>

<requires>

A list of required files, such as HTML pages, CSS stylesheets or JavaScript libraries, to load with this app.

Can Contain	"<require>" on page 1314 +
Allowed In	<app>

<require>

One file (HTML, CSS or JavaScript) that must be loaded with this app, with optional metadata to control loading behavior.

Can Contain	Empty
Allowed In	<app> or <requires>

Attributes

Name	Required	Description
type	yes	The type of file to load: ■ `html` = an HTML page or fragment for this app. ■ `css` = a CSS stylesheet that has *not* been configured in MashZone NextGen for automatic dependency loading. ■ `script` = a JavaScript library that has *not* been configured in MashZone NextGen for automatic dependency loading. ■ `library` = a JavaScript library that is configured in MashZone NextGen for automatic dependency loading. Configured libraries, also known as *named libraries*, can contain any number of JavaScript or CSS files and declare any JavaScript dependencies.
src	conditional	For all files that are *not* named MashZone NextGen libraries (type != `library`), this is the relative or absolute URL to the file to load with this app. Relative URLs must point to files within the app package (see "App Packages and App Files" on page 1318 for more information). They *cannot* use paths that redirect up the folder tree past the app's root folder, such as ../above-root/myLibrary.js.
name	conditional	For named MashZone NextGen libraries (`type = 'library'`), the name of the library to load.

Name	Required	Description
		Important: MashZone NextGen includes certain "*third party software*" which JackBe licenses from third parties. Pursuant to the JackBe EULA for MashZone NextGen, all MashZone NextGen users are bound by the license terms and conditions of any third party software licenses. You may review these Third Party Licenses at http://documentation.softwareag.com/legal/. Software AG does not and cannot authorize any use of third party software that is not permitted by these third party software licenses. You may, however, be able to obtain licenses directly from third parties. In addition, any other software that you add and/or use in connection with MashZone NextGen is subject to its own licensing requirements and may void the terms of the EULA for MashZone NextGen.
version	conditional	Identifies a version of this app resource. The specific value and meaning of the resource version depends on the value of the `type` attribute: ■ For `type = 'library'` (named MashZone NextGen libraries), this idenfies the specific version of the named library to load. If you omit the version, the current version is loaded. ■ For `type != 'library'` (any other file), you can use this attribute to ensure that browsers load updated files rather than cached versions. When you update an app resource, change this attribute to a number greater than the previous version number, if any. The MashZone NextGen App Framework adds this number to the URL for the resource which ensures that browsers download the updated resource rather than using a cached file. **Tip:** It is a good practice to use simple integers as version numbers for app resources.
loadconfirmation		A boolean expression containing one or more JavaScript namespaces or objects from this library

Name	Required	Description
		that must exist at runtime to confirm that this library is completely loaded. For example: `MyOrg && MyOrg.Config` **Important:** Because the App Spec is XML, the JavaScript & operator must be escaped using `&`, the XML escape for this character. This expression is not required but is *highly* recommended to ensure that libraries for the app are properly loaded before rendering.

<title>

An optional title for the app. In many environments, this title displays in a header bar above the app content.

Can Contain	Text.
Allowed In	<app>

<topics> or <topic>

<topics>

A list of topics communicated between apps that this app either subscribes to or publishes. See "Declare App Topics and Payloads" on page 1285 for examples.

Can Contain	"<topic>" on page 1316 *
Allowed In	<app>

<topic>

One topic communicated between apps that this app either subscribes to or publishes. This element enables wiring with other apps for this app and topic.

Can Contain	"<description>" on page 1303?, "<properties>" on page 1310

Allowed In	<app>

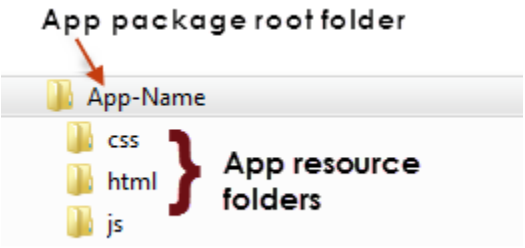
Attributes

Name	Required	Description
name	yes	The name of this topic. **Tip:** It is a best practice to start topic names with a unique namespace, such as the namespace used by the custom app that subscribes to or publishes the topic. Names for topics *must* be unique within an app. You may use any character that is valid in JavaScript, however, keep the following rules in mind: ■ Both `.` and `*` have a specific meaning in topic names. Topic names are composed of one or more tokens and `.` is the token separator. You can use `*` as a wildcard for individual tokens in topic names when you subscribe to topics. So for a subscription, both `client.*.issue` and `myOrg.**` are valid topic names. ■ Do not use the following characters: ■ White space characters, such as space or tab ■ Quote marks or other punctuation, such as ? or (■ Control characters For more information on valid topic names, see the OpenAjax Hub 2.0 Specification.
datatype		The datatype that describes the payload for messages in this topic. Valid datatypes include: ■ `string` ■ `number` ■ `boolean` ■ `date`

Name	Required	Description
		■ `timestamp` ■ `object` = a complex payload. This is the default. ■ `any` = any type is accepted. This datatype has three implications to keep in mind: ■ No <properties> should be specified for topics with this datatype since there are no limits. ■ Because no properties are specified, wiring can only happen at the topic level. You can wire a published topic to a subscription topic with a datatype of `any` but you cannot wire properties in that published topic. ■ The handler function for subscription topics with this datatype must be able to handle messages with inappropriate data.
publish		Whether this app publishes messages to this topic (`true` | `false`). Defaults to `false`. **Note:** Only one of `publish` or `subscribe` can be true for a topic.
subscribe		Whether this app subscribes to messages for this topic (`true` | `false`). Defaults to `false`. **Note:** Only one of `publish` or `subscribe` can be true for a topic.

App Packages and App Files

A custom app is defined by a set of files that you add to the MashZone NextGen Repository using the App Editor in MashZone NextGen Hub. These files are organized in an *App package* with this base folder structure:



This basic app package includes the following app files:

File Type	Default File Name	Description
App Specification	app.xml	This is an XML file that defines configuration and meta-data for the app. See the "App Specification Reference" on page 1298 for links to information on the elements you use in an App Spec.
HTML	html/ app.html	This can be a full HTML page or just a fragment, such as a , that renders the user interface for this app. It can have any file name.
	index.html	This page is not required when users run apps, but is part of the standard package for custom app development. It allows you to run your custom app as you work with the code to test it within your own development environment and tools.
JavaScript libraries	js/app.js	This JavaScript file contains the code that you write specifically for this app.
	other libraries as needed	You may include any other JavaScript libraries in an app including: ■ Any JavaScript file within the folder tree of your app. You use a relative URL for these files. Relative paths *cannot* redirect up the folder tree. ■ Common JavaScript frameworks or plug-in libraries that are hosted in MashZone NextGen including prototype, jQuery and many others. You simply identify these frameworks and libraries by name. MashZone NextGen handles all dependencies for named libraries.

File Type	Default File Name	Description
		■ Other JavaScript frameworks or libraries hosted externally or internally using an absolute URL.
CSS	css/app.css	This CSS file contains the styles you define for your custom app. You can include other CSS files in any folder within the app's folder tree using a relative URL. Or you can include CSS files hosted externally or internally using an absolute URL.
Images		The standard development package for a custom app does not include any image files. You can include images for this app in any folder within the app's folder tree using a relative URL.
Other		Files for other media or protocols as needed by your app. Like images, files for other media must reside in folders within the app package and use a relative URL.

You can get a starting app package by starting a new app in the App Editor and downloading this. See "Create Custom Apps from the Base App Package" on page 1226 for more information. Or you can download the app package for an existing app that you have opened in the App Editor.

In most cases, you update the app package and individual app resources in your own development environment, then zip up these updates and use this to update the app in MashZone NextGen. You can, however, add or remove files for apps in the App Editor. See "Working in the App Editor" on page 1194 for more information.

MashZone NextGen administrators can also manage files for apps from the Admin Console.

12 MashZone NextGen Add-Ons

- The MashZone NextGen Add-On for Excel 1322
- The MashZone NextGen Add-On for Portals 1341
- MashZone NextGen Add-On for SharePoint (P4S) 1358
- MashZone NextGen Add-On for SharePoint User Guide 1445

MashZone NextGen Add-Ons allow you to work with non-traditional information sources, such as spreadsheet documents. Add-ons can also be destinations for mashups and apps, allowing users in your organization to work with these artifacts in applications or environments they are already comfortable with. For an overview of these add-on components, see "MashZone NextGen Add-Ons" on page 63 in the *Introduction*.

MashZone NextGen Add-Ons include the:

- **MashZone NextGen Add-On for Excel**: which allows you to:
  - Import mashable or mashup data into Excel worksheets to work with this data.
  - Register snapshots of worksheet data or register dynamic links to worksheet data from Excel as spreadsheet mashables in MashZone NextGen.

  See "Install the MashZone NextGen Toolbar in Excel" on page 1323 for installation instructions. See "The MashZone NextGen Add-On for Excel" on page 1322 for information on using this add-on.
- **MashZone NextGen Add-On for Portals**: which allows you to easily add MashZone NextGen mashups and apps to pages in your portal.

  For information and links, see "The MashZone NextGen Add-On for Portals" on page 1341 .
- **MashZone NextGen Add-On for SharePoint**: which allows you to work with SharePoint lists and searches in MashZone NextGen mashups and apps. You can also import mashups or apps into native SharePoint web parts to combine with existing web parts or wire together.

  For information and links, see "MashZone NextGen Add-On for SharePoint (P4S)" on page 1358.

The MashZone NextGen Add-On for Excel

The MashZone NextGen Add-On for Excel allows you to work with data in Excel and MashZone NextGen from end to end. This add-on includes the "MashZone NextGen Toolbar for Excel" on page 1324 which you can add to Excel and use to :

- Import results from MashZone NextGen information sources, *mashables* and *mashups*, to work with this information in Excel.
- Export data in your worksheets as MashZone NextGen mashables so that other users can work with your information in mashups or apps.

MashZone NextGen makes it easy to get information from many different sources such as databases in your organization, public information available on the Internet, information from partner organizations or information that other users in your organization have put together and made available as a mashable information source or mashup.

To start working with the MashZone NextGen Add-On for Excel, see "Install the MashZone NextGen Toolbar in Excel" on page 1323 for instructions. Then configure your connection to MashZone NextGen. See "Configuring the Connection to MashZone NextGen" on page 1325.

Once the toolbar is installed and configured, you can begin using the MashZone NextGen Toolbar to Work with MashZone NextGen Mashup and Mashable Results in Excel or Export Excel Worksheets or CSV Files as Mashables in MashZone NextGen.

Install the MashZone NextGen Toolbar in Excel

The MashZone NextGen Add-On for Excel installs an add-on toolbar in Excel that you can use to export worksheet data from Excel as a MashZone NextGen mashable to share with other users, use in mashups or create apps. You can also use this toolbar to import results from MashZone NextGen mashables or mashups into an Excel worksheet and get updates to these results.

Before You Start

You must have the following software installed:

- Microsoft Office 2007 or Office 2010 (32-bit)

  Note: Because of a known issue, the MashZone NextGenMashZone NextGen Add-On for Excel is not currently compatible with Microsoft Office 2010 for 64 bit systems.

- Microsoft .NET Framework 2.0 or above
- Office 2007 Primary Interop Assemblies

Installation

1. Download the MashZone NextGen Add-On for Excel installation package from the Software AG Tech Community for MashZone NextGen at http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto/.
2. Run the ExcelPrestoConnectSetup.msi file to start the MashZone NextGen Add-On for Excel installation wizard.
   a. Click **Next** once the Welcome Page opens in the installation wizard.
   b. If needed, change the default folder where you want to install the MashZone NextGen Add-On for Excel.

      This defaults to c:\Program Files\JackBe\MashZone NextGen Connect Excel Connector or c:\Program Files (86)\JackBe\MashZone NextGen Connect Excel Connector.

   c. Click **Next** twice to confirm your installation choices.

Installation begins.

d. Once installation is complete, click **Close**.

3. Open Excel to verify that the MashZone NextGen toolbar is available in the Add-Ons tab:

MashZone NextGen Toolbar for Excel

The MashZone NextGen Toolbar for Excel lets you import mashup and mashable data from MashZone NextGen to an Excel worksheet or export data from an Excel worksheet as a mashable information source to give other users access or to work with worksheet data in mashups or apps.

Button	Description
Import Mashable/ Mashup	Imports the results from one operation of a MashZone NextGen mashup or mashable into the current worksheet.
Set Input Parameters	Sets values for the input parameters to send to a MashZone NextGen mashup or mashable to obtain results for the current worksheet.
Refresh Imported Results	Invokes a MashZone NextGen mashup or mashable to refresh the results.
Remove Imported Mashable/Mashup	Removes configuration for an imported MashZone NextGen mashup or mashable from the current worksheet. This does not remove any data.
Register Worksheet in MashZone NextGen	■ Registers a range in the current worksheet or any accessible worksheet file to MashZone NextGen as a new spreadsheet mashable. ■ Updates an existing spreadsheet mashable in MashZone NextGen with changes or additions in a range of the current worksheet or any accessible worksheet file.
Remove Spreadsheet Mashable	Removes a spreadsheet mashable information source in MashZone NextGen and disconnects the

Button	Description
	relationship between MashZone NextGen and the worksheet.
Configure Connection	Sets connection information to MashZone NextGen for importing, registering or updating mashable information sources.
Online Help	Help information for the MashZone NextGen Add-On for Excel.

If you do not have the MashZone NextGen Toolbar in Excel, see "Install the MashZone NextGen Toolbar in Excel" on page 1323 for instructions on how to add this to Excel.

Note: Your MashZone NextGen administrator can also hide or show the MashZone NextGen Toolbar using administration tools.

The MashZone NextGen Toolbar Add-On Configuration File

Installing the MashZone NextGen Toolbar for Excel adds a configuration file, the MashZone NextGen Add-On for Excel configuration file, that keeps track of all the information needed to connect to the MashZone NextGenMashZone NextGen Servers that you import data from or export spreadsheets to. This file is located at either:

Windows XP or Windows Vista	C:\Documents and Settings*your user name* \Application Data\PC4Excel\config.xml
Windows 7	C:\Users*your user name* \AppData\Roaming\PC4Excel \config.xml

The configuration in this file is updated automatically. In rare cases, you may need to update this manually for connections that use HTTPS. See "Configuring the Connection to MashZone NextGen" on page 1325 for details.

When you uninstall the MashZone NextGen Toolbar for Excel you can choose to leave this file in place or allow it to be deleted. If you are uninstalling your current version of the MashZone NextGen Toolbar for Excel in order to install a newer version, this allows you to keep your existing configuration information.

Configuring the Connection to MashZone NextGen

You define connection information for MashZone NextGen that is used when you import mashup or mashable results from MashZone NextGen, when you register

worksheets as spreadsheet mashables in MashZone NextGen or when you update spreadsheet mashables with updates from your worksheet.

To configure the connection

1. Click *Configure Connection*.

   The configuration window opens with default information to connect to a MashZone NextGen Server running on your computer.

2. Enter the URL to the MashZone NextGen Server you want to use to import, register or update mashable information sources.

   The URL should have a host name or an IP address and a port, such as `http://127.0.01:8080`. Contact your MashZone NextGen administrator for MashZone NextGen URLs.

   For connections to the MashZone NextGen Server that use HTTPS and SSL, your computer typically has a personal certificate already installed to support secure connections and no additional configuration is needed. If your organization uses *self-signed* certificates, however, you must:

   a. Open the MashZone NextGen Add-On for Excel configuration file, config.xml in Notepad or any simple text editor. See the "The MashZone NextGen Toolbar Add-On Configuration File" on page 1325 for the appropriate location of this file.

      This file is in an XML format that should look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<Configurator xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <ConfigFileName>C:\Documents and Settings\Some User\Application
    Data\PC4Excel\config.xml</ConfigFileName>
  <ShowToolbar>true</ShowToolbar>
  <CultureId>en-US</CultureId>
  <WebProxyHost />
  <WebProxyPort>0</WebProxyPort>
  <WebProxyUser />
  <WebProxyPassword />
  <WebProxyEnabled>false</WebProxyEnabled>
  <WebProxyBypassLocalAddresses>false</WebProxyBypassLocalAddresses>
  <Timeout>1</Timeout>
  <UseSelfSignedCertificate>false</UseSelfSignedCertificate>
  <ServerList>
```

   b. Find the line with `<UseSelfSignedCertificate>false</UseSelfSignedCertificate>` and change the value to `true`.

      The file should now look like:

```
<?xml version="1.0" encoding="utf-8"?>
<Configurator xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <ConfigFileName>C:\Documents and Settings\Some User\Application
   Data\PC4Excel\config.xml</ConfigFileName>
  <ShowToolbar>true</ShowToolbar>
  <CultureId>en-US</CultureId>
  <WebProxyHost />
  <WebProxyPort>0</WebProxyPort>
  <WebProxyUser />
  <WebProxyPassword />
  <WebProxyEnabled>false</WebProxyEnabled>
  <WebProxyBypassLocalAddresses>false</WebProxyBypassLocalAddresses>
  <Timeout>1</Timeout>
  <UseSelfSignedCertificate>true</UseSelfSignedCertificate>
  <ServerList>
```

c. Save your changes to this file.

3. Enter your **User name** and **Password** for MashZone NextGen.

   You can click **Test** at this point to determine if Excel can connect to the MashZone NextGen Server. If needed, set proxy server configuration before you test the connection.

4. Optionally, change the **Timeout** period. Enter the number of minutes that Excel should continue trying to connect to the MashZone NextGen Server.

5. If you use a proxy server to connect to the MashZone NextGen Server, define proxy server configuration in the **Proxy Settings** tab:

   a. Set the **Enable Web Proxy Settings** option.

   b. In **Address**, enter the host name or number for the proxy server.

   c. Enter the **Port** number for the proxy server.

   d. If the proxy server requires you to log in, enter the **User name** and **Password** for the proxy server.

   e. If you only use the proxy server to connect to remote MashZone NextGen Servers but you also sometimes connect to a local MashZone NextGen Server, set the **Bypass web proxy for local addresses** option.

      With this option set, Excel will use the proxy server only when you import, register or update to remote MashZone NextGen Servers.

6. Click **OK** to save this configuration.

Work with MashZone NextGen Mashup and Mashable Results in Excel

MashZone NextGen mashups and mashables provide easy access to a wealth of information from many different sources that you can import into a worksheet. This can include data from databases in your organization, information from other applications in your organization, information from public sources through the Internet or from partner organizations and information that has been combined or transformed in a mashup.

To use mashup or mashable results in a worksheet, you Import MashZone NextGen Mashup or Mashable Results. You can also Refresh MashZone NextGen Mashup or Mashable Results or Change Parameters for an Imported Mashup or Mashable to obtain new information . For more information, see also "Changing Imported Mashup or Mashable Data" on page 1333 and "Remove an Imported Mashup or Mashable" on page 1333.

Import MashZone NextGen Mashup or Mashable Results

You can import results from MashZone NextGen mashups or mashables into Excel worksheets to work with this information in your workbooks.

To import mashup or mashable results into an Excel worksheet

1. If needed, set connection information to the MashZone NextGen Server for the mashable information source or mashup that you want to import. See "Configuring the Connection to MashZone NextGen" on page 1325 for instructions.
2. Move to the worksheet and cell that should be the upper-left corner where results should be imported.

   Tip: To prevent space or modification conflicts, it is a good practice to separate MashZone NextGen mashup or mashable data into a different worksheet from the calculations, charts or other data you are working with in a workbook. If you import data from several mashups or mashables, it is also best to import results side by side (into separate sets of columns).

3. Click *Import Mashable/Mashup*.

   The Import Mashable Wizard opens.

4. Click **Next**.

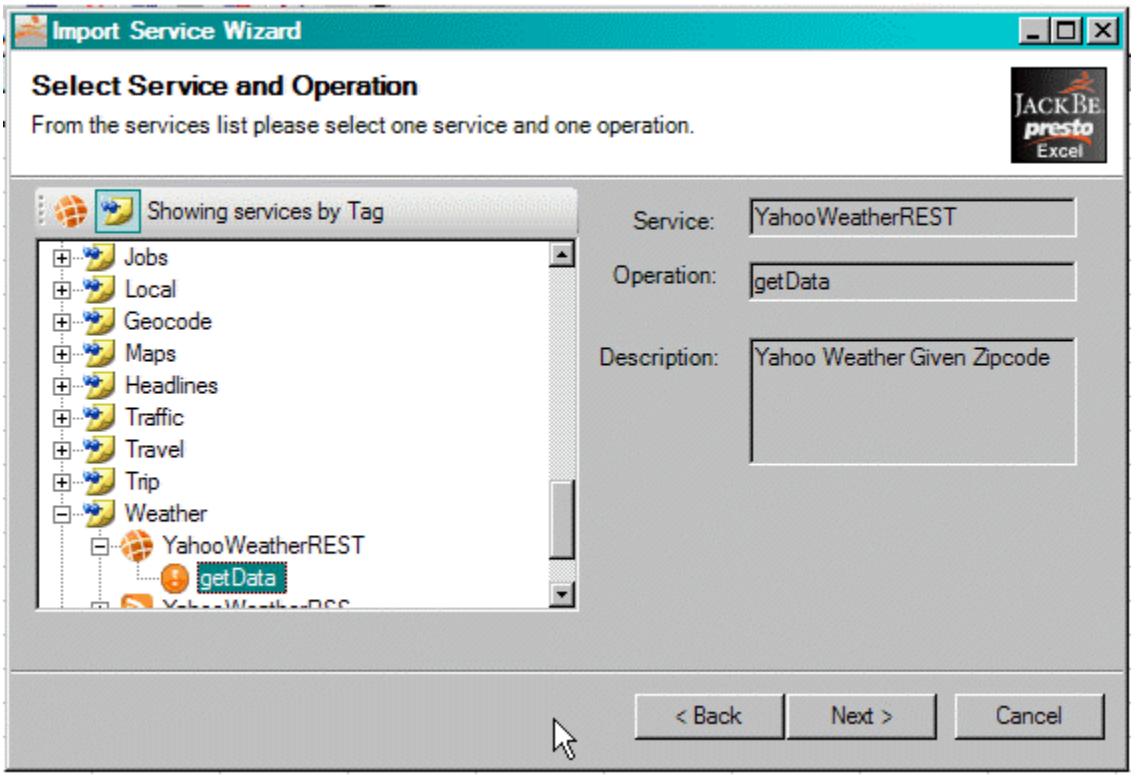
   The Select Mashable and Operations screen displays a list of MashZone NextGen mashups and mashables categorized by *mashable type*, such as Atom/RSS or Excel. You can also sort this list by tag categories, such as News or Finance.

   Note: The icon indicates mashables or mashups that are inactive. You cannot import data from inactive mashables or mashups.

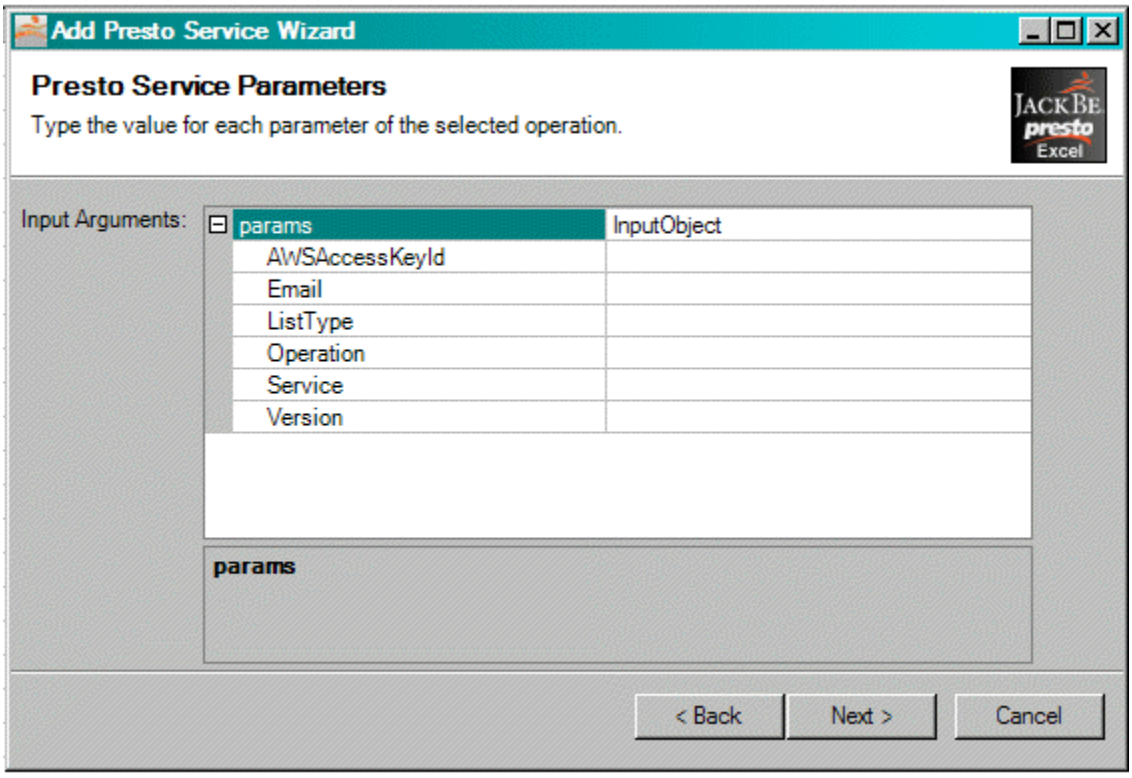
5. If desired, change the search categories for mashups and mashables:
   - Click in the list header to organize mashables and mashups **By Tag**.
   - Click in the list header to organize mashables and mashups **By Type**.

     See "Types of Mashable Information Sources" on page 110 for more information on mashable types.

   Expand any category and then expand a mashable or mashup to see the list of operations available.



6. Select the operation that you want to run and import the results. Click **Next**.
7. If this operation has input parameters that you can set, expand the parameters list, fill in the parameters as needed and click **Next**.

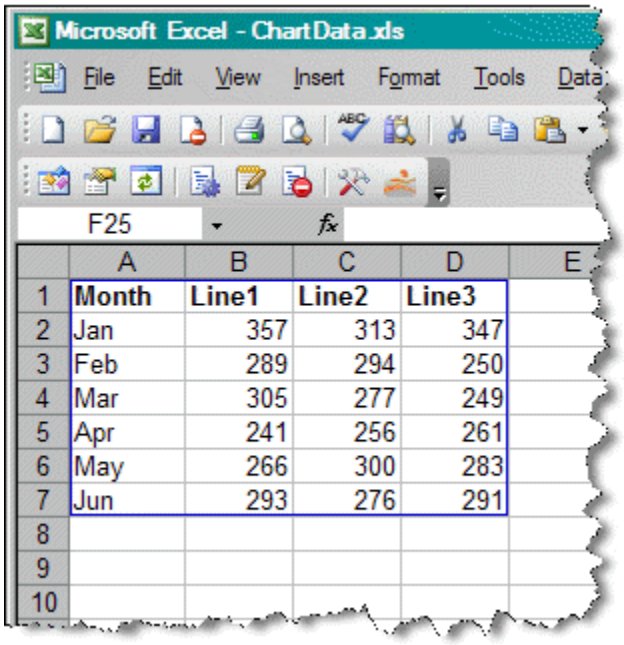


The wizard runs the mashup or mashable operation and displays the structure of the results in a tree in the next pane.The top node containing all the results is selected by default.

8. If you only want to import part of the results, select the node that contains all the information that you want to import. See "Selecting the Result Nodes to Import" on page 1331 for more information and examples on selecting specific information and the results.
9. Click **Next**.

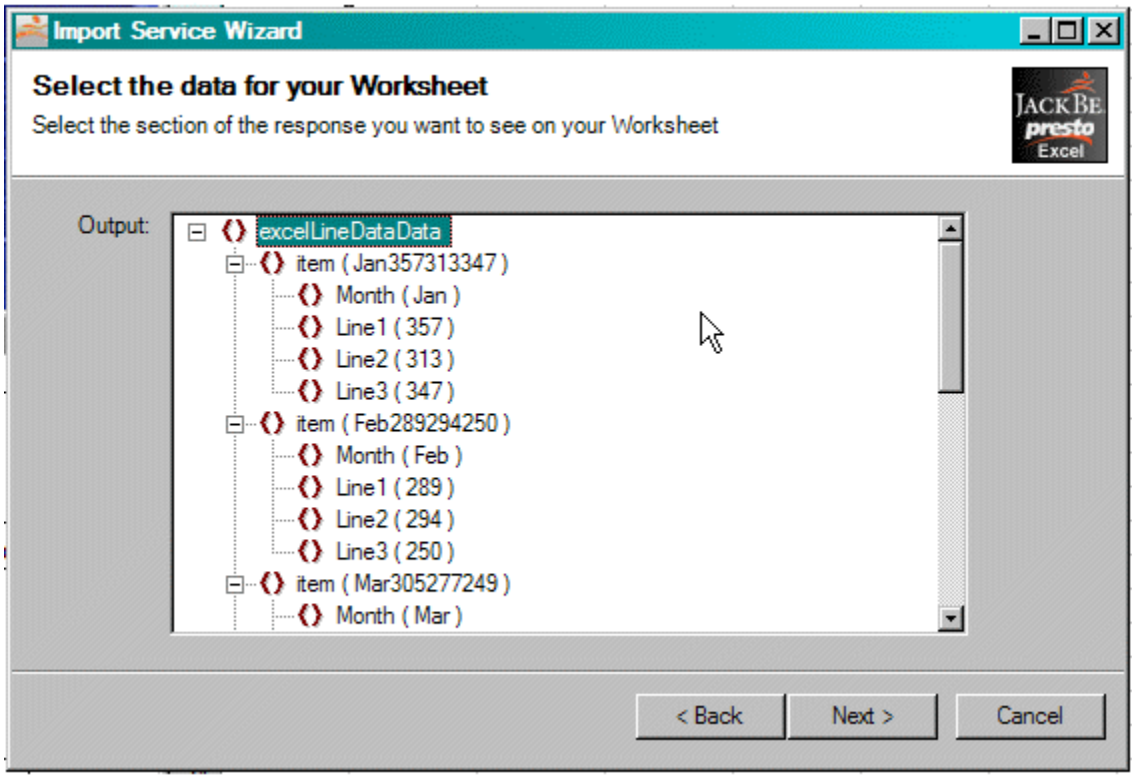
   The wizard displays a summary of your import choices.
10. Click **OK** to complete the import.

A range of cells, outlined in blue, fill with results from this mashable operation.



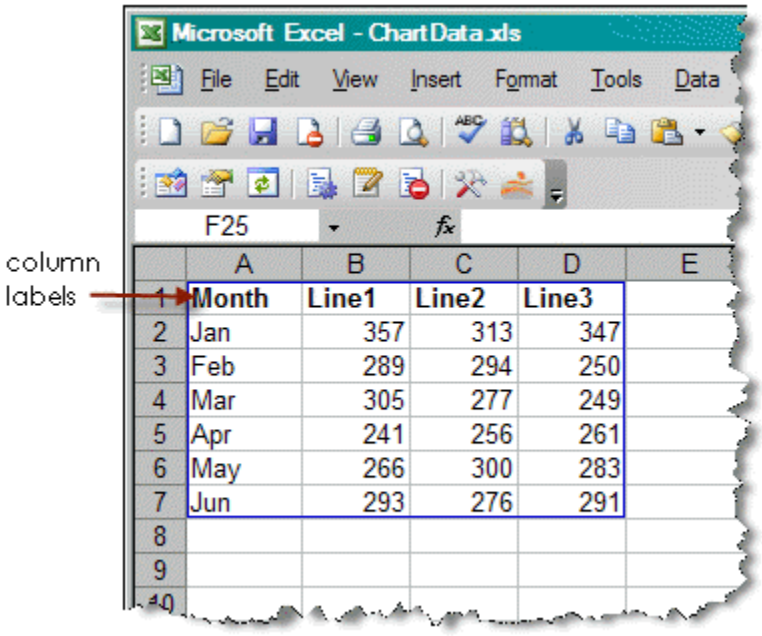
Selecting the Result Nodes to Import

When you import mashup or mashable results, you must select what information to import from a tree showing you the structure of the result, such as this:



By default, the top node (sometimes called the root node) is selected. This will import the entire results into a grid of rows and columns.

The results from many mashups and mashables have a set of repeating *items* that are imported into Excel as separate rows. Each item has children nodes with data that appear in Excel as different columns. Importing the information can also provide *column labels* to help identify what the information is about.



If you do not want to import all the results. you can select different nodes within the sample response tree. If you want to import:

- *All Items in a List of Items*: select the parent node that above the entire list of items. Selecting the parent nodes ensures that each column of information has a column label.
- *One Item in a List*: select that item only. Only the data for that item will be imported into one row. No column labels will be imported.
- *One Specific Node*: select a specific node. Only the data for that node will be imported into a single cell. No column label will be imported.

Change Parameters for an Imported Mashup or Mashable

You can change the initial parameters you used to run a MashZone NextGen mashup or mashable and then refresh the results in your worksheet.

To change the parameters

1. Click *Edit Mashable Parameters*.

2. Select the imported mashup or mashable that you want to update from the **Imported Mashables** list.

   The list of parameters for this operation displays with their current values.

3. Edit the parameters values and click **Save** once they are correct.

Changing Imported Mashup or Mashable Data

Once you have imported data from a MashZone NextGen mashup or mashable information source, you can use this in calculations or other worksheets within your workbook. You can also modify mashup or mashable data, keeping these suggestions in mind:

- *Changing column labels* to make them more meaningful is recommended. Limit your column names to ASCII letters, numbers, periods (.), dashes (-) or underscores (_). Do not use spaces or other symbols.
- *Deleting rows* except for the first row with column labels is okay. The next time you refresh mashup or mashable data, however, the data you deleted returns.
- *Changing data* in individual cells is not recommended, although Excel will not stop you from doing this. The next time you refresh mashup or mashable data, the data you changed is overwritten.

Refresh MashZone NextGen Mashup or Mashable Results

To get updated results for a mashup or mashable information source that you have imported into your worksheet, simply refresh the results. You can also change parameter values and then refresh results.

To refresh results

1. If needed, update the parameters for imported mashables or mashups. See "Change Parameters for an Imported Mashup or Mashable" on page 1332 for instructions.
2. Click *Refresh Mashable Results*.

Remove an Imported Mashup or Mashable

Removing an imported mashup or mashable information source removes the configuration information in your worksheet to connect to MashZone NextGen and refresh data for this mashup or mashable. It does *not* remove mashup or mashable data.

To remove imported results

1. Click **Remove Imported Mashable**.
2. Select the imported mashup or mashable information source that you want to remove from this worksheet and click **Remove**.
3. When prompted, click **OK** to remove the import configuration.

4. Delete the imported data using Excel.

Export Excel Worksheets or CSV Files as Mashables in MashZone NextGen

In addition to importing information from MashZone NextGen, you can also make information in worksheets or CSV files available in MashZone NextGen as spreadsheet mashables. As a spreadsheet mashable, other users can use this data in mashups or apps. Or import your information into Excel for their own use.

When you register a worksheet with the MashZone NextGen Toolbar, the speadsheet mashable can have either a snapshot of the information from your worksheet or a dynamic link to your worksheet. See "Manual or Automatic Updates to Spreadsheet Content" on page 1334 for more information. See "Valid Spreadsheet Mashables" on page 1339 for information on the format and content that you can include in a mashable.

You can also Update Mashables Manually from Worksheets to make updates to your worksheet available to other users. Or disconnect a worksheet from the mashable and Remove an Excel Mashable from MashZone NextGen.

Register a Worksheet as a Spreadsheet Mashable in MashZone NextGen

You can publish information in an Excel workbook by registering a spreadseheet mashable in MashZone NextGen. This makes the information available for use in mashups, to distribute in apps or to share with other users as a mashable source that they can import to their own worksheets. You can:

- Register a Range of Cells From the Current Worksheet
- Register a Worksheet From an Excel File in Excel
- Register a CSV File as a Mashable

You can publish any amount of Excel or CSV data to the MashZone NextGen Server. Size limitations for data present in previous releases of the MashZone NextGen Add-On for Excel no longer apply. See also "Valid Spreadsheet Content" on page 1334 and "Manual or Automatic Updates to Spreadsheet Content" on page 1334.

Valid Spreadsheet Content

Some worksheet content is *not* available in the spreadsheet mashable. See "Valid Spreadsheet Mashables" on page 1339 for information on what worksheets or cell ranges can and cannot include.

Manual or Automatic Updates to Spreadsheet Content

The MashZone NextGen mashables registered for spreadsheets can contain a snapshot of the current worksheet data, or they can have a dynamic link to the spreadsheet that

always returns updated data. This depends on the access to your worksheet when you register the mashable:

- If your worksheet is hosted in SharePoint or can be accessed remotely using HTTP, MashZone NextGen gets a dynamic link to your worksheet. Any updates to your worksheet are automatically reflected in MashZone NextGen.
- If access to the worksheet is local, from your local computer or a network drive, the spreadsheet mashable gets a snapshot of the data. Any updates you make to the worksheet do not automatically update the spreadsheet mashable in MashZone NextGen. You must Update Mashables Manually from Worksheets.

Register a Range of Cells From the Current Worksheet

If needed, set connection information to the MashZone NextGen Server where you want to register a mashable for this worksheet. See "Configuring the Connection to MashZone NextGen" on page 1325 for instructions. See "Register a Worksheet as a Spreadsheet Mashable in MashZone NextGen" on page 1334 for basic information about registering mashables for worksheets.

1. Open the workbook and worksheet you want to register as a mashable.
2. Click *Register/Update a Spreadsheet Mashable* in the MashZone NextGen Toolbar.
3. Click **Next** in the welcome window of the Register Wizard.
4. In the Choose Data Source window:
   a. Set the *From a Range in the Current Worksheet* option.
   b. Click **Select Range**.
   c. In the worksheet, select the range of cells you want to export.
   d. In the Wizard, click **Done** once the range of cells is correct.
   e. Click **Next**.
5. Select the option to use for column names in the spreadsheet mashable:
   - *Use first row of range as column names* is the default.
   - **Pick a row number for column names** set this option and enter the number of the row within your range of cells that contains column names.
   - *Select a range for column names* set this option and click **Select Columns**. In the worksheet, select a range of cells within *one* row that you want to use as column names and click **Done**.

   If any cell in this row is empty, the MashZone NextGen Add-On for Excel will generate simple column names, such as Column1, Column2 and so on.
6. Click **Next**.
7. Review the range selected in the Data Preview window and click **Next** if this is correct.

8. Complete information for the mashable you are publishing:
   a. Enter a unique **Name** for this mashable.

      MashZone NextGen uses the mashable name to assign a unique identifer to mashables. Mashable names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

   b. Optionally, select a **Provider** for this mashable, add **Tags** or enter a **Description** for this mashable.

      You can also use the built-in provider: `NONE`.

   c. Click **Next**.
9. Click **OK** once all information is correct.

   Excel confirms that the mashable is successfully registered.

Register a Worksheet From an Excel File in Excel

If needed, set connection information to the MashZone NextGen Server where you want to publish this mashable. See "Configuring the Connection to MashZone NextGen" on page 1325 for instructions. See "Register a Worksheet as a Spreadsheet Mashable in MashZone NextGen" on page 1334 for basic information.

To register a worksheet from an Excel file in Excel

1. Click *Register/Update a Spreadsheet Mashable* in the MashZone NextGen Toolbar.
2. Click **Next** in the welcome window of the Register Wizard.
3. Set the **From Excel File** option and choose the file location.

   If your workbook is located on your computer or on a network drive:

   a. Set the *Local Excel File Location* option.

   b. Click **Browse** and locate the Excel file that you want to use.

   c. Enter the worksheet and **Range** of cells to use in the form:

      sheet-name `$`*corner-cell* `:`*corner-cell*

      For example: `Sheet1$A1:G348`

   d. Click **Next**.

   If the workbook is in SharePoint or a document management system and is available using HTTP and a URL:

   a. Set the *Remote Excel File Location* option.

   b. Enter the full URL to the worksheet you want to use.

   c. If SharePoint or the document management system requires credentials to access this workbook:

- Click **Credentials**.
- Change the **Authentication Type** to `Basic Web Authentication` or `Windows Authentication`.
- Enter the user name and password to use as credentials.
- For Windows authentication, enter the Windows domain for these credentials.

d. Enter the worksheet and **Range** of cells to use in this worksheet in the form:

sheet-name `$`*corner-cell* `:`*corner-cell*

For example: `Sheet1$A1:G348`

e. Click **Next**.

4. Select the option to use for column names in the spreadsheet mashable:
   - *Use first row of range as column names* is the default.
   - **Pick a row number for column names** set this option and enter the number of the row within your range of cells that contains column names.
   - *Select a range for column names* set this option and click **Select Columns**. In the worksheet, select a range of cells within *one* row that you want to use as column names and click **Done**.

   If any cell in this row is empty, the MashZone NextGen Add-On for Excel will generate simple column names.
5. Click **Next**.
6. Complete information for the mashable you are registering:
   a. Enter a unique **Name** for this mashable.

   MashZone NextGen uses the mashable name to assign a unique identifer to mashables. Mashable names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`

   b. Optionally, select a **Provider** for this mashable, add **Tags** or enter a **Description** for this mashable.

   You can also use the built-in provider: `NONE`.

   c. Click **Next**.
7. Click **OK** once all information is correct.

   Excel confirms that the mashable is successfully registered.

Register a CSV File as a Mashable

If needed, set connection information to the MashZone NextGen Server where you want to publish this mashable. See "Configuring the Connection to MashZone NextGen" on page 1325 for instructions.

You can also use comma-separated-value (CSV) files to provide data for MashZone NextGen mashables.

To register a CSV file as mashable

1. Click *Register/Update a Spreadsheet Mashable* in the MashZone NextGen Toolbar.
2. Click **Next** in the welcome window of the Register Wizard.
3. Set the **From local CSV file** option and.
   a. Click **Browse** and locate the CSV file that you want to use.
   b. Select the character that is used as the separator between fields in this CSV file:

Comma	This (,) is the default separator.
Pipe	| is also known as the upright bar symbol.
Tab	Another common separator.
Other	For any other separator character. Then enter the actual separator character in the open field.

   c. Clear the **CSV file has column names** option if the first row of the CSV file does not contain column names.
4. Review the data in the Data Preview window and click **Next** if this is correct.
5. Complete information for the mashable you are registering:
   a. Enter a unique **Name** for this mashable.

      MashZone NextGen uses the mashable name to assign a unique identifer to mashables. Mashable names can contain characters from the character sets supported by theMashZone NextGenRepository, numbers, spaces, tabs, line ends and these common symbols: `_ ~ - * ' .`
   b. Optionally, select a **Provider** for this mashable, add **Tags** or enter a **Description** for this mashable.

      You can also use the built-in provider: `NONE`.
   c. Click **Next**.
6. Click **OK** once all information is correct.

Excel confirms that the mashable is successfully registered.

Valid Spreadsheet Mashables

Workbooks or cell ranges should be limited to the following to create valid MashZone NextGen mashables:

- **Formulas** in spreadsheets are valid, but the formula itself is not exported to the mashable in MashZone NextGen. The result of the calculation (the data) is exported instead.
- **Column labels**:
  - Are recommended, but optional, for each column that contains data.

    If column labels are missing, MashZone NextGen automatically generates labels in the form `Column1, Column2` and so on.
  - Are limited to letters, numbers and underscores in the ISO-8859-1 character set, also known as Latin 1.

    Spaces and most symbols or punctuation characters are either converted to an underscore or ignored.
  - Must start with a letter (an alphabetical character) although they can contain numbers after that. Column labels that do not start with a letter are altered or overridden.
  - If a column label starts with a number or is a formatted date, MashZone NextGen adds a prefix of `Column_` to the label.
  - Can only be provided by *one* row. For CSV files, this must be the first row.

    For spreadsheets you can identify which row supplies the column labels when you register the mashable. If your spreadsheet has column names in two or more rows, you must choose a single row, allow MashZone NextGen to generate column labels or simplify your worksheet.
  - Cannot span two or more cells in a spreadsheet. MashZone NextGen treats the second and subsequent columns as missing labels. You can allow MashZone NextGen to generate column names for the second and subsequent columns or you can update your worksheet.
- *Row labels* are not supported. These should be treated as a column of data and have a column label.
- *Hidden columns* are included when you register a mashable in MashZone NextGen.
- **Charts, images or other objects** in spreadsheets are ignored and not saved in mashable data.

Update Mashables Manually from Worksheets

When your worksheet is on your local computer or a networked drive, updates to your Excel worksheets do not automatically update any MashZone NextGen spreadsheet

mashables that you have registered with the worksheet. You must manually update the MashZone NextGen mashable information source with changes to your worksheet.

Note: Restrictions from previous releases that required linking worksheets to Excel mashables in MashZone NextGen are no longer applicable.

To update mashables from a worksheet manually

1. Update your worksheet, as needed, and save your changes.
2. Click *Register/Update a Spreadsheet Mashable* in the MashZone NextGen Toolbar and click **Next**.
3. Choose the source of the updated data:
   - *From a Range in the Current Worksheet*, click **Select Range**.

     In the worksheet, select the range of cells with updates for this mashable. In the Wizard, click **Done** once the range of cells is correct.
   - *From Excel File*, choose **Local Excel File Location** for worksheets on your local computer or on a networked drive. Then:
     - Click **Browse** and locate the Excel file with updates.
     - Enter the worksheet and **Range** of cells with updates in the form:

       sheet-name `$`*corner-cell* `:`*corner-cell*

       Note: Updates to worksheets in SharePoint or another document management system happen automatically, sothe Remote Excel File option is not relevant.
   - **From local CSV file**, click **Browse** and locate the CSV file with updates for this mashable.

     Change the separator character or column name options, if needed.
4. Set the **Update Spreadsheet Mashable** option and click **Next**.
5. For worksheets, select the option to use for column names:
   - *Use first row of range as column names* is the default.
   - **Pick a row number for column names** set this option and enter the number of the row within your range of cells that contains column names.
   - *Select a range for column names* set this option and click **Select Columns**. In the worksheet, select a range of cells within *one* row that you want to use as column names and click **Done**.

   If any cell in this row is empty, the MashZone NextGen Add-On for Excel will generate simple column names.
6. Click **Next**.

7. Complete information for the mashable you want to update:
   a. Enter the **Name** for this mashable.
   b. Optionally, change the **Provider**, **Tags** or **Description** for this mashable.
   c. Click **Next**.
8. Click **OK** once all information is correct.

   Excel confirms that the mashable is successfully updated.

Remove an Excel Mashable from MashZone NextGen

If needed, set connection information to the MashZone NextGen Server with the mashable information source that you want to remove.

You can remove a mashable definition in MashZone NextGen for your current Excel worksheet. This has no affect on the worksheet. It simply removes the mashable information source in MashZone NextGen and breaks the link to the current worksheet.

To remove an Excel mashable

1. Click **Remove Spreadsheet Mashable** in the MashZone NextGen Toolbar.
2. Click **Next** in the welcome pane of the Remove Spreadsheet Mashable Wizard.
3. Select the spreadsheet mashable you want to remove from MashZone NextGen and click **Next**.
4. Review the summary information and click **OK** if this is correct.

The MashZone NextGen Add-On for Portals

The MashZone NextGen Add-On for Portals is a *portlet producer* that you can add to your portal applications to add MashZone NextGen mashups and apps to portal pages as portlets. This MashZone NextGen add-on is available for any portal that supports Java Portlet standards.

See "Installing the MashZone NextGen Add-On for Portals" on page 1342 for instructions for new installations. For upgrades to new releases, see "Migrating to New Versions of the MashZone NextGen Add-On for Portals" on page 1347.

See the "MashZone NextGen Add-On for Portals Architecture" on page 1345 for a basic introduction to the MashZone NextGen Add-On for Portals.

Once you have installed the MashZone NextGen Add-On for Portals, you add portlets for mashups or apps to your portal pages. See "Adding MashZone NextGen Mashups to Portal Pages" on page 1349, "Adding MashZone NextGen Mashups to Portal Pages" on page 1349 and "Adding MashZone NextGen Apps to Portal Pages" on page 1351 for instructions.

For information on updating the size of portlets, updating input parameters for apps or mashups and many other visual aspects that you can control with portlet preferences, see "Managing and Updating Mashups or Apps in Portal Pages" on page 1353.

Installing the MashZone NextGen Add-On for Portals

The MashZone NextGen Add-On for Portals allows you to quickly create portlets for any MashZone NextGen mashup or app and publish them to portal pages. You can also add any mashable information source to portal pages by wrapping it with a simple mashup in MashZone NextGen.

These instructions cover deploying the MashZone NextGen Add-On for Portals locally in the same application server as your portal for new installations. For upgrades to an existing version of the MashZone NextGen Add-On for Portals, see "Migrating to New Versions of the MashZone NextGen Add-On for Portals" on page 1347.

You can deploy the MashZone NextGen Add-On for Portals remotely, but you must generate the WSRP file and deploy that instead. See the documentation for your portal for more information on remote deployments.

To install the MashZone NextGen Add-On for Portals

1. Download the MashZone NextGen Add-On for Portals
2. Download and Install MashZone NextGen
3. Add the MashZone NextGen Add-On for Portals to Your Portal
4. Update Configuration for the MashZone NextGen Add-On for Portals
5. Login and Get Started

Download the MashZone NextGen Add-On for Portals

Download the MashZone NextGen Add-On for Portals installation package from the Software AG Tech Community for MashZone NextGen at http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto/ and extract the contents of the Zip file to any folder on your computer.

Note: To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com.

This contains the MashZone NextGen Add-On for Portals web application, presto-portlets.war.

Download and Install MashZone NextGen

If you have not already done so, download and install MashZone NextGen. See *Installing Software AG Products* for instructions. Start the MashZone NextGen Server.

Apply the latest fixes, if any, to MashZone NextGen using the Software AG Update Manager. See *Using the Software AG Update Manager* for instructions.

Add the MashZone NextGen Add-On for Portals to Your Portal

Typically, you first deploy the MashZone NextGen Add-On for Portals WAR file in your application server. See documentation for your application server for more information on this step.

Important: For JBoss, WebLogic or any application server that does not fully expand WAR file content for web applications, you *must* update connection information to the MashZone NextGen Server during deployment. Typically this means you cannot use auto-deploy features, but must deploy the MashZone NextGen Add-On for Portals WAR file manually using a command line, the administration console or other administrator tools for your application server.

See the "Update Configuration for the MashZone NextGen Add-On for Portals" on page 1343 section for information on the context parameters you should update.

Then, register the MashZone NextGen Add-On for Portals as a *portlet producer* in your portal. See documentation for your portal for instructions on adding portlet producers.

Update Configuration for the MashZone NextGen Add-On for Portals

Once the MashZone NextGen Add-On for Portals is registered, there are two places where you may need to update configuration before you start working with the MashZone NextGen Add-On for Portals:

Connection Information to the MashZone NextGen Server

The MashZone NextGen connection configuration for the MashZone NextGen Add-On for Portals defaults to the local host using the default TomEE port. If you used a different port, you must update the connection configuration to the correct URL.

If your application server fully expands web application WAR files, edit the web.xml configuration file for the MashZone NextGen Add-On for Portals in *web-apps-home* / presto-portlets/WEB-INF.

For application servers that do not expand WAR files, you must update configuration properties when you deploy the MashZone NextGen Add-On for Portals.

Update these `<context-param>` configuration properties:

- *jackbe.presto.server*: when your portal has a single, well-known connection to one MashZone NextGen Server, simply update the default URL in the `<param-value>` element. This defaults to `http://localhost:8080/presto`.

  If your portal has *virtual portals* that each connect to a different MashZone NextGen Server, you can define dynamic connections based on the URL to each virtual portal. With dynamic connections, you enter the static portions of the URL along with

tokens indicating those portions that must be supplied dynamically in this property. For example:

```
http://$1.portal.$2:10000/presto
```

The tokens `$1` and `$2` are replaced by capture groups defined in a regular expression in the `jackbe.presto.serverMatchPattern` configuration property.

- *jackbe.presto.serverMatchPattern*: use this context parameter when your portal has several *virtual portals* that each should connect dynamically to different MashZone NextGen Servers. In this case, the URL to each MashZone NextGen Server must be derived dynamically based on the URL to a specific virtual portal.

  You define the static portions of each MashZone NextGen Server URL in the `jackbe.presto.server` configuration property. You use this property to define a regular expression that defines *capture groups* for the dynamic portions of the URL.

  In the `<param-value>`, enter the regular expression that should be applied to each virtual portal URL to provide the dynamic values to derive the URL to connect to the appropriate MashZone NextGen Server. For more information on capture groups in regular expressions, see Capture Groups in Java Patterns documentation.

  For example, two virtual portal URLs have host and domains in the form `red.portal.myorg.com` and `blue.portal.myorg.com`. Using the regular expression `^(\w+)\.(.*)$` results in two tokens: `red` and `myorg.com` for virtual portal 1 and `blue` and `myorg.com` for virtual portal 2.

  If you combine this with the static URL and tokens shown previously, this connects virtual portal 1 to `http://red.portal.myorg.com:10000/presto` and virtual portal 2 to `http://blue.portal.myorg.com:10000/presto`

Save your changes.

Java Memory Allocations for Your Application Server

You may also need to increase the Java memory parameters in startup scripts or other initialization files for the portal. The minimum recommendations are:

- `-Xm512M`
- `-Xmx1024M`
- `-XX:PermSize=256M`
- `-XX:maxPermSize=512M`

Login and Get Started

If you have changed memory or connection configuration, restart your application server and portal to apply these changes. If you have not already done so, start the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for more information.

Log in to MashZone NextGen using the default administrator credentials: `Administrator` (username) and `manage` (password). The default URL is http://localhost:8080/presto/.

Once you have some mashups and apps, login to your portal and add either the `Publish Mashups` or `Publish Apps` portlets to portal pages to add mashups and apps to your portal. See "The MashZone NextGen Add-On for Portals" on page 1341 for more information and links.

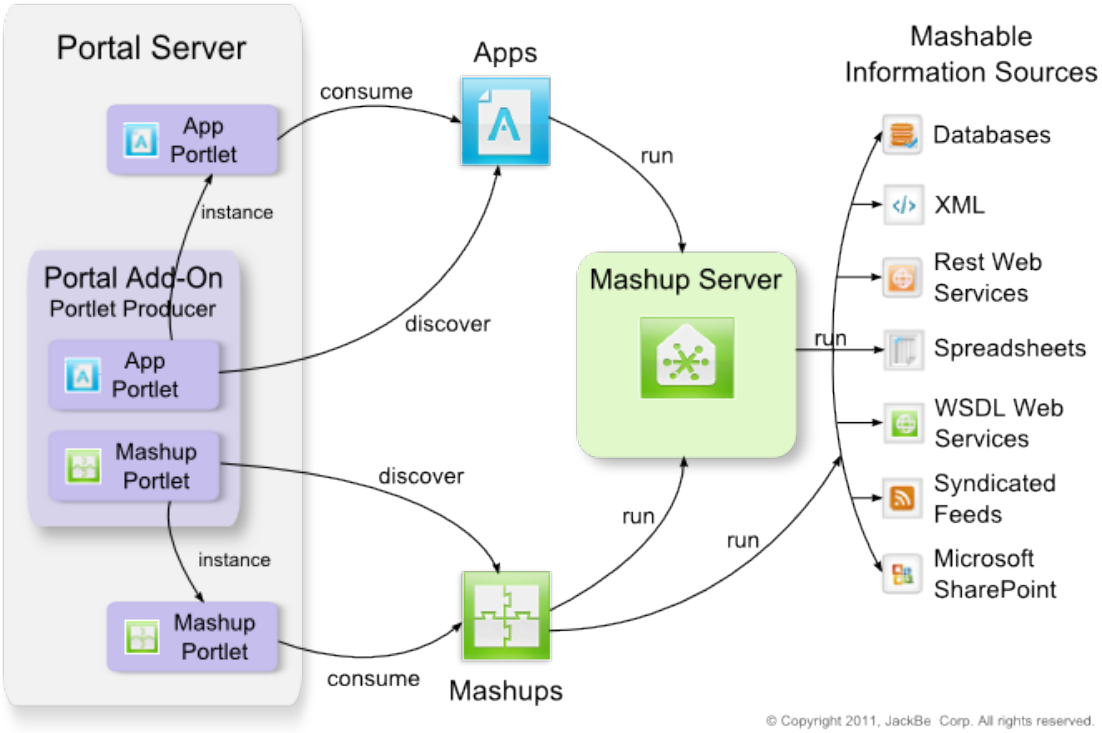
MashZone NextGen Add-On for Portals Architecture

Integrating MashZone NextGen mashables, mashups and apps into a portal environment provides several benefits:

- A simple, but powerful way to consume many common information sources in your portal, including web services, web feeds, relational databases or common business documents
- A simple, but powerful mashup capability to filter, combine and transform information to fit your unique needs
- A simple way to govern access to information provided by MashZone NextGen artifacts (mashables, mashups and apps)

The MashZone NextGen Add-On for Portals simplifies the integration process between your portal application and MashZone NextGen. You plug the add-on into your portal application and quickly begin working with MashZone NextGen mashups and apps.

The MashZone NextGen Add-On for Portals is compatible with portals that support Java Portal standards. This add-on can be hosted with your portal server or it can be hosted separately, if the portal server supports WSRP. The following figure shows the basic architecture for this connector:



Portlets

The MashZone NextGen Add-On for Portals is a *portlet producer* which you integrate with your portal application. You use the MashZone NextGen Add-On for Portals to work with two portlets:

- *Publish Mashup Portlet*: to add MashZone NextGen mashups to portal pages (as portlet instances). You can also work with MashZone NextGen mashable information sources using this portlet by creating simple *wrapper* mashups that run any mashable information source.
- *Publish App Portlet*: to add MashZone NextGen apps and workspace apps to portal pages (as portlet instances).

These portlets allow you to discover MashZone NextGen mashups and apps, add them to portal pages and configure them for users. You can also allow users to add portlets, discover MashZone NextGen artifacts or simply customize portlet configuration to meet their own needs.

Authentication and Authorization for MashZone NextGen Artifacts

Authentication for portal users accessing MashZone NextGen artifacts in a portlet depends on how authentication has been configured in MashZone NextGen and your portal. Possible solutions include:

- *Single Sign-On* solution. For SSO, the portlet request to MashZone NextGen will have information on an existing SSO session which MashZone NextGen uses to authenticate the portlet user.

  See "Authentication with Single Sign-On Solutions" on page 1607 for more information.

- *SSL/Digital Certificates*. For SSL, the portlet request to MashZone NextGen will have included the user's digital certificate which MashZone NextGen uses to authenticate the portlet user.

  See "Authentication with Digital Certificates/SSL" on page 1614 for more information.

- *Username/password credentials*. This is the default authentication scheme for MashZone NextGen.

  For user credential authentication, the user must have already logged into MashZone NextGen in their browser to create a valid MashZone NextGen session. MashZone NextGen uses this session information to authenticate the user.

If authentication information is available when portlets request MashZone NextGen apps or mashups, portlet users must have run permissions in MashZone NextGen to view and use the specific MashZone NextGen app or mashup that is configured for that portlet.

If no authentication information is available, MashZone NextGen considers the user a guest. For apps, the specific app used in the portlet must have run permissions that allow guest access. Guest access to mashups, however, is not supported.

Migrating to New Versions of the MashZone NextGen Add-On for Portals

New versions of the MashZone NextGen Add-On for Portals are compatible with the matching version of MashZone NextGen. Upgrading to a new release typically involves upgrading both MashZone NextGen and the MashZone NextGen Add-On for Portals. The basic steps are:

1. Download the New MashZone NextGen Add-On for Portals Release
2. Upgrade MashZone NextGen to the Matching Release
3. Backup Portal Configuration
4. Install the New MashZone NextGen Add-On for Portals Release
5. Clear the Portlet Cache in the Portal

Download the New MashZone NextGen Add-On for Portals Release

Download the MashZone NextGen Add-On for Portals installation package for the new release from the Software AG Tech Community for MashZone NextGen at http://

techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto/ and extract the contents of the zip file to any folder on your computer.

This contains the MashZone NextGen Add-On for Portals web application, presto-portlets.war, for this release.

Upgrade MashZone NextGen to the Matching Release

If you have not already upgraded MashZone NextGen to the same version as the new release of the MashZone NextGen Add-On for Portals:

1. Download this MashZone NextGen release from Empower, the Software AG support portal, at http://empower.softwareag.com/ and upgrade MashZone NextGen.
2. Install the new MashZone NextGen release. See *Installing Software AG Products* for instructions.
3. Complete any upgrade steps needed. See *Upgrading Software AG Products* for instructions.
4. Then apply the latest fixes, if any, to MashZone NextGen using the Software AG Update Manager. See *Using the Software AG Update Manager* for instructions.

Backup Portal Configuration

Before applying the new MashZone NextGen Add-On for Portals release, make backups of any updates to configuration for the existing MashZone NextGen Add-On for Portals in your current installation. This typically includes the *web-apps-home* /presto-portlets/WEB-INF/web.xml file. See any recommended practices for your portal for more information.

Install the New MashZone NextGen Add-On for Portals Release

Redeploy the MashZone NextGen Add-On for Portals web application, presto-portlets.war, for this new release following the recommended redeployment practices for your portal.

Then update configuration for the MashZone NextGen Add-On for Portals using the configuration backups saved earlier.

Clear the Portlet Cache in the Portal

If your portal caches portlets, you must also clear this cache before you begin working with the new MashZone NextGen Add-On for Portals release. See documentation for your portal for instructions on this process.

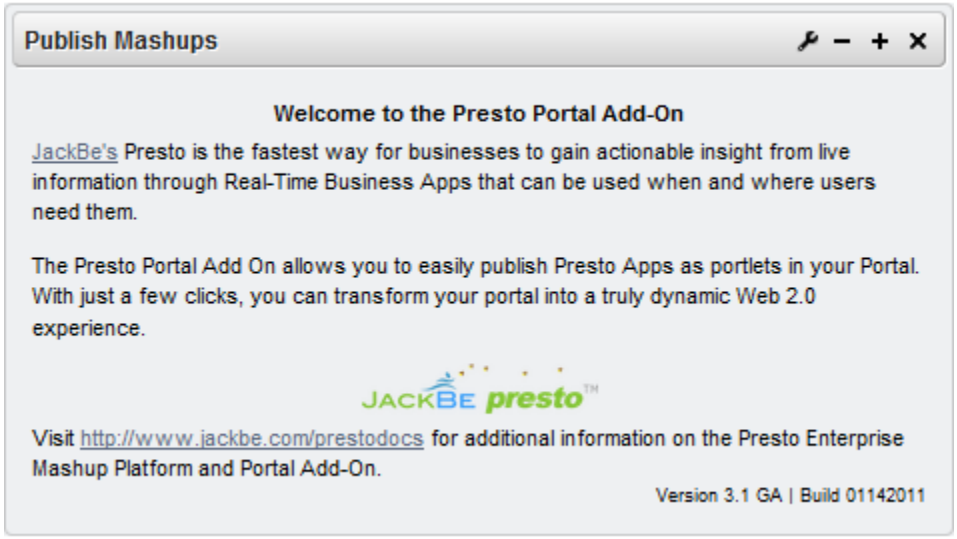
Finally, check for specific migration tasks in the MashZone NextGen Release Notes, if any, to update portlets for this release.

Adding MashZone NextGen Mashups to Portal Pages

You create mashups and add views for them in the MashZone NextGen Hub. Once mashups have at least one view, you can then add them to portal pages. You can also add any mashable information source by creating a mashup that simply runs that mashable.

Note: New views that are added to mashups that are already included in portal pages automatically appear in those portal pages.

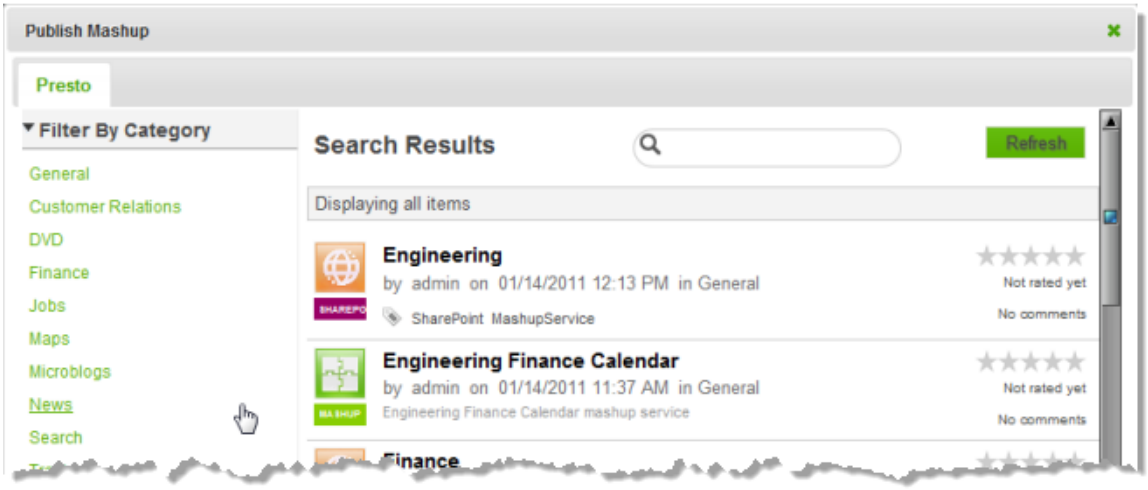
1. In your portal or portal IDE, start a new portal page or edit an existing page and add the *Publish Mashup* portlet.
   a. If needed, save the changes to this page.
   b. If it is not already open, log in to your portal and navigate to the page where you added this portlet. The portlet opens initially with the help page and should look something like this:



2. Click the **Configure** button in the portlet title bar and select the option to set preferences or customize the portlet.

   Note: The icons, buttons and menu options may be different in your portal.

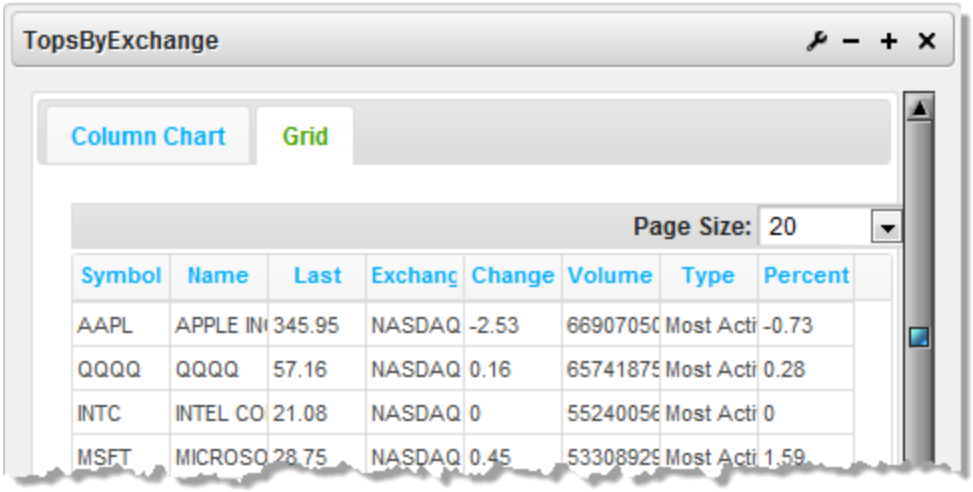
3. Find and select the mashup you want in the gallery window that opens.



You can filter the list using the Category or Provider links. Or enter part of the name, a tag or term in the Search field to find mashups by name, tag or description.

4. If desired, update the default input parameters, if any, for this mashup.
5. If needed, configure the layout for this portal page and drag the mashup portlet to the appropriate area within the page.
6. Click **Finish**.

   The mashup opens in the portlet with the default view for this mashup.



All views that are defined for this mashup appear in the default `Stacked` layout. You can change this layout or update other visual aspects of the portlet or input parameters, if any, for the mashup in the portlet preferences. See "Managing and Updating Mashups or Apps in Portal Pages" on page 1353 for instructions.

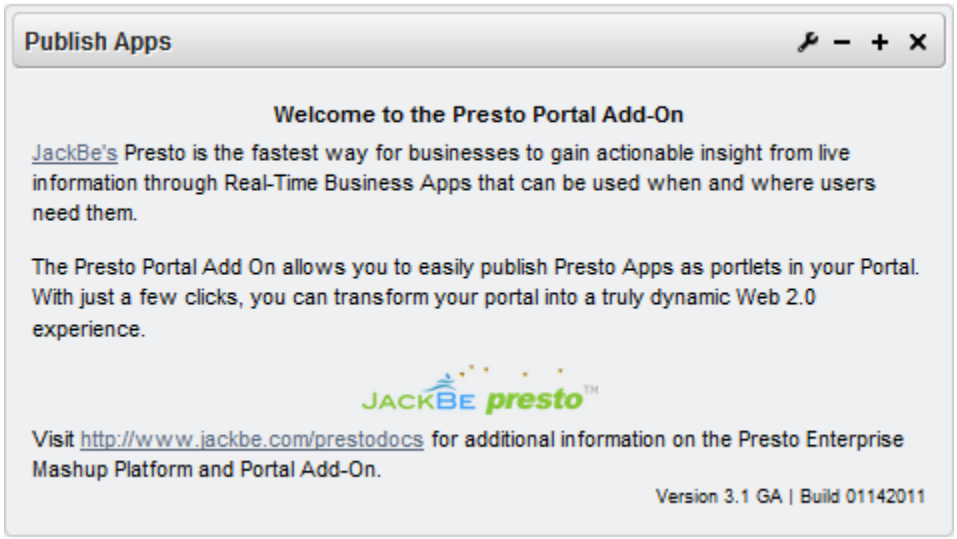
Adding MashZone NextGen Apps to Portal Pages

You create apps and workspace apps in the MashZone NextGen Hub using mashable information sources, mashups or gadgets from other environments. Once you have apps defined in MashZone NextGen, you can add them to portal pages.

Note: Changes to MashZone NextGen apps that have been included in portal pages automatically update the portal pages.

See also "Managing and Updating Mashups or Apps in Portal Pages" on page 1353 for information on preferences and updates for apps.

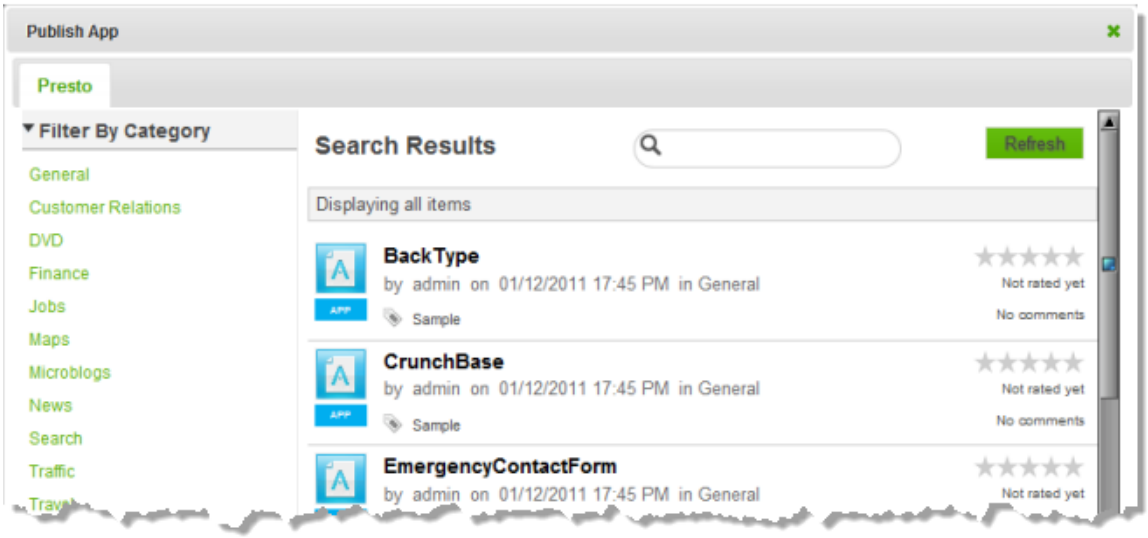
1. In your portal or portal IDE, start a new portal page or edit an existing page and add the *Publish Apps* portlet.
   a. If needed, save the changes to this page.
   b. If it is not already open, log in to your portal and navigate to the page where you added this portlet. The portlet opens initially with the help page and should look something like this:



2. Click the **Configure** button in the portlet title bar and select the option to set preferences or customize the portlet.

   Note: The icons, buttons and menu options may be different in your portal.

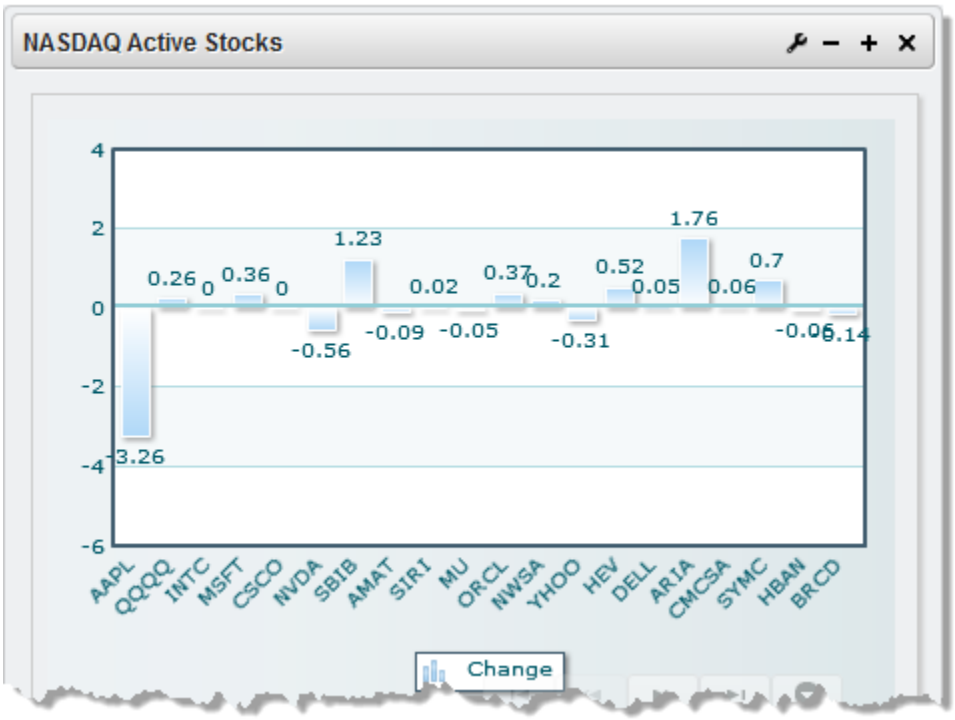
3. Find and select the app you want in the gallery window that opens.



You can filter the list using the Category or Provider links. Or enter part of the name, a tag or term in the Search field to find apps by name, tag or description.

4. If desired, update the default input parameters, if any, for this app.
5. If needed, configure the layout for this portal page and drag the app portlet to the appropriate area within the page.
6. Click **Finish**.

   The app opens in the portlet.



With the layout and theme assigned to the app in MashZone NextGen. By default, the app title bar, view titles and input form for app input parameters, if any, are hidden.

Users can change app input parameters in portlet preferences. You can also show the app title bar or update other visual aspects for the app portlet in preferences.

Managing and Updating Mashups or Apps in Portal Pages

You can use any of the portlet configuration options supported by your portal. This generally includes:

- Updating the page layout, fonts, colors or other visual aspects of the page.
- Deleting portlets from a page.
- Moving portlets to new positions within the page layout.
- Setting user permissions and many others.

Updates to the underlying mashups or apps in MashZone NextGen automatically update the portal pages where they are included. If the underlying mashups or apps are deleted from MashZone NextGen, portal pages that include these artifacts will display errors that they cannot be found.

In addition to the basic options for your portal, you can also use "Portlet Preferences" on page 1354 to update different aspects of mashups or apps in your portal pages.

Portlet Preferences

If you have permissions or are the page owner, you can set default preferences for mashup and app portlets. Other users may also choose to customize these preferences for their own use based on their permissions.

To update preferences, click the **Configure** button in the portlet title bar or edit toolbar and select Preferences or Default Preferences. With preferences you can:

- Change the Display Size for a Mashup or App
- Change Mashup or App Input Parameters
- Enable the App Forms for Input Parameters, Sorting and Filtering
- Show the App Title Bar
- Change the Portlet Title
- Update the Mashup Layout for Views
- Update Mashup or App Pagination Settings
- Update App Theme
- Update App Portlet Performance Preferences

Change the Display Size for a Mashup or App

Open preferences for the mashup or app portlet. In the Configure Portlet Settings section, update these preferences:

- **Height**: enter the specific height, in pixels, to display this mashup or app.
- **Width**: enter `auto` to have the mashup or app completely fill the width of the current column in this portal page. Or enter the specific width, in pixels, to display this mashup or app.

Save your changes.

Change Mashup or App Input Parameters

Open preferences for the mashup or app portlet. Update any input parameters in the Parameters section and save your changes.

For apps, you may also be able to update input parameters, sorting criteria or filtering criteria in the respective forms for the app if these forms are enabled and the app title bar is also enabled. If enabled, any user can open these forms at any time using the button in the app title bar.

See also "Enable the App Forms for Input Parameters, Sorting and Filtering" on page 1355 and "Show the App Title Bar" on page 1355.

Enable the App Forms for Input Parameters, Sorting and Filtering

Users who have permissions to update preferences for an app portlet can always change the input parameters for an app. You can also choose to show the default MashZone NextGen forms for input parameters, sorting criteria and filtering criteria for the app to allow *any user* to change these parameters or conditions for their current session.

Note: Apps may have none or any combination of these forms available for users.

There are two preferences that control access to app forms: **Hide Input Form** and **Hide App Header**. Both of these preferences are disabled by default.

To allow users access to all of the app's forms:

1. Open preferences for the portlet.
2. Move to the Configure Portlet Settings section.
3. Set **Hide Input Form** to `false`.

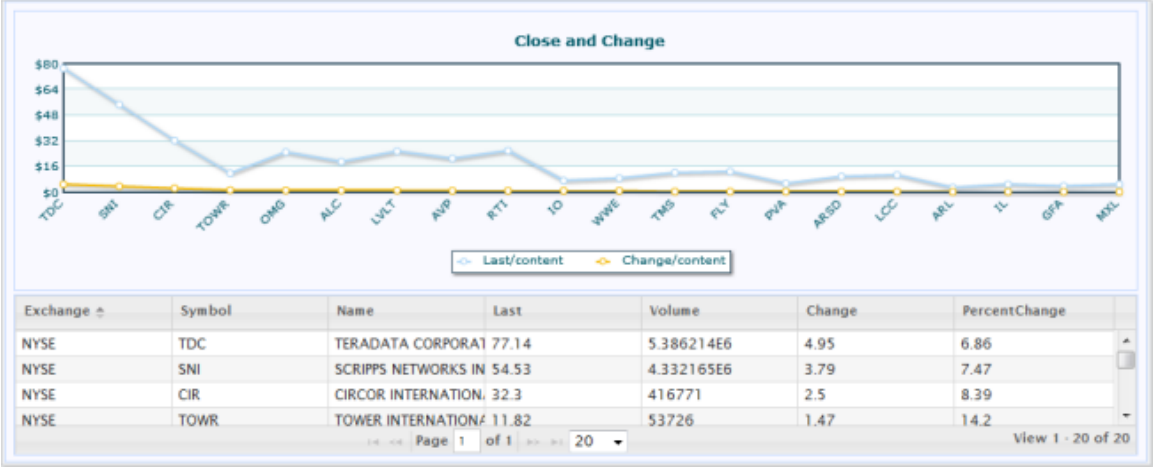
   This enables all of the forms for the app, not just the form for input parameters.
4. Enable the app's title bar (see "Show the App Title Bar" on page 1355) to make the toolbar button visible with access to these forms.
5. Save your changes.

Show the App Title Bar

In MashZone NextGen, apps can have a title bar, with a toolbar, and title bars for each view included in the app such as this example:

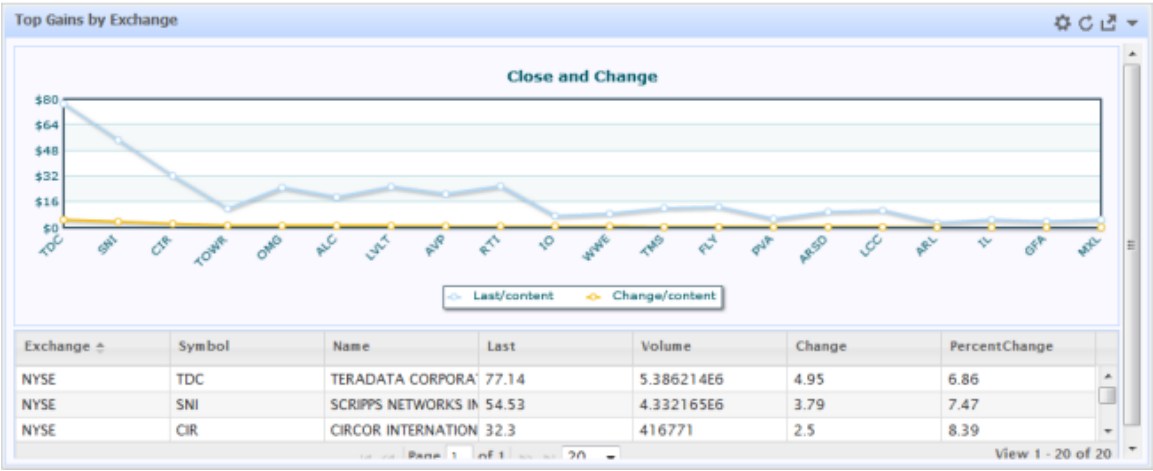
In the app portlet, both the app title bar and view title bars are hidden by default such as this next example:

Default App portlet, App Title Bar hidden



View title bars are always hidden, but you can show the app title bar in the portlet, as this example illustrates:

App portlet with App Title Bar visible



This title bar displays the app title and includes a toolbar with buttons that allow any user to refresh the app, update input parameters, sorting criteria or filtering criteria for the app or open the app in a new window or tab.

To show the app title bar:

1. Open preferences for the portlet.
2. Move to the Configure Portlet Settings section.
3. Set **Hide App Header** to `false`.
4. Save your changes.

You may also want to enable the toolbar button with access to input, sorting and filtering forms. See "Enable the App Forms for Input Parameters, Sorting and Filtering" on page 1355 for instructions.

Change the Portlet Title

The portlet title appears in the Preferences/Edit title bar for the owner or other users who have permissions to edit preferences for a portlet. This title defaults to the app or mashup name from MashZone NextGen. You can update the portlet title in preferences for the portlet:

1. Open preferences for the portlet.
2. Move to the Configure Portlet Settings section.
3. Enter the new title in the **Title** preference.
4. Save your changes.

Update the Mashup Layout for Views

When you add a mashup to a portal page, all the views that are defined in MashZone NextGen are shown in a vertical list using the default `Stacked` layout. You can change this layout if desired:

1. Open preferences for the portlet.
2. Move to the Configure Portlet Settings section.
3. Choose the **Layout** you want to use:
   - `Stacked` = stack views vertically. This is the default layout.
   - `Tabbed` = show each view in its own tab. Note: this was the only layout available in previous releases.
   - `Two Column` = list views in two columns stacked vertically
   - `Wide Top Row` = list view in two columns stacked vertically, but the first row contains one view that spans both columns.
4. Save your changes.

Update Mashup or App Pagination Settings

You can also change the number of records that are shown in one page of information for all views in an app portlet.

1. Open preferences for the app portlet.
2. In the Configure Portlet Settings section, update the **Rows per Page** preference with the number of records to show in one 'page' of data for all views in the app.
3. Save your changes

Update App Theme

You can change the borders and background colors for an app with the **Theme** preference. If the app Title Bar is shown, this also affects the background color of the title bar.

To update the theme, open preferences for the app portlet. In Edit Settings, choose a new theme color and save these changes.

Update App Portlet Performance Preferences

By default MashZone NextGen apps are added to portal pages using an <iframe> element. Iframes ensure that any CSS styling or JavaScript used within the app portlet does not conflict with other portal content in the page.

In some cases where several apps appear in a portal page, iframes can affect performance making the portal page slow. You can choose to add app portlets directly, known as *inline rendering*. This may improve performance but may also generate CSS or JavaScript conflicts.

To render app portlets inline in the portal page:

1. Open preferences for the portlet.
2. Move to the Configure Portlet Settings section.
3. Enter `true` in the **Inline** preference.
4. Save your changes.

MashZone NextGen Add-On for SharePoint (P4S)

The MashZone NextGen Add-On for SharePoint (P4S) can be a:

- Source of information for MashZone NextGen mashables or mashups, combining or working with SharePoint information from many different SharePoint sites or farms.
- Source of external information in SharePoint lists using the Business Connectivity Service (BCS). This allows mashup results from the information sources that MashZone NextGen connects to to provide dynamic data to SharePoint lists.
- Published destination for MashZone NextGen apps and mashups.

  You use this add-on to create native Web Parts for apps or mashups, providing dynamic, real-time integration to the full range of information sources in MashZone NextGen. This also leverages MashZone NextGen governance for these resources.

Note: The MashZone NextGen Add-On for SharePoint is only available if it is enabled in your MashZone NextGen license.

For a full introduction to P4S, see the "An Introduction to the MashZone NextGen Add-On for SharePoint" on page 1360. For links to topics on working with this add-on in SharePoint or MashZone NextGen, see the following sections.

Working with the MashZone NextGen Add-On for SharePoint in SharePoint

For information on working with P4S to publish apps or mashups as Web Parts or connect mashups as sources for external content, see:

- "An Introduction to the MashZone NextGen Add-On for SharePoint" on page 1360
- "Publishing Apps and Mashups to Web Parts" on page 1416
- "Publish Mashup Results as Native SharePoint Lists" on page 1427
- "Publish Mashup Results as External SharePoint Lists" on page 1428
- "Register a SharePoint List as a MashZone NextGen Mashable" on page 1434 in SharePoint.

Working with SharePoint in MashZone NextGen Hub

For information on working with P4S to use SharePoint as an information source for MashZone NextGen mashables, mashups or apps, see:

-
- "Register SharePoint Lists" on page 156 as mashables
-
- "Add SharePoint Information Sources" on page 290 to mashups in Wires
-
- "Publish an App in SharePoint" on page 1102 in MashZone NextGen Hub
- "SharePoint Credentials for Mashables and Mashups" on page 1411

MashZone NextGen and SharePoint Administrators

SharePoint and MashZone NextGen administrators should also see:

- "Installing the MashZone NextGen Add-On for SharePoint Solution" on page 1367
- "P4S Configuration and Administration, SharePoint 2010" on page 1372
- "P4S Configuration and Administration, SharePoint 2007" on page 1392
-
- "Configure MashZone NextGen Connections to SharePoint" on page 1655
- "SharePoint Credentials for Mashables and Mashups" on page 1411

An Introduction to the MashZone NextGen Add-On for SharePoint

The MashZone NextGen Add-On for SharePoint (P4S) brings the power of the Software AG MashZone NextGen to your existing SharePoint environment. This powerful product enables a dynamic two-way workflow between MashZone NextGen and SharePoint:

- MashZone NextGen to SharePoint: users can rapidly build, use, and share mashups and apps for use in SharePoint. Dynamic information can be shared as Web Parts for display or more deeply integrated as external lists using the Business Connectivity Service.
- SharePoint to MashZone NextGen: SharePoint lists from one or several sites can be seamlessly used in your mashups and apps and combined with information from other sources or transformed.

Thus, P4S enhances the SharePoint experience by empowering business users to create mashups and apps that combine and display diverse, real-time data sources, including SharePoint lists, CRM, ERP, relational databases, and RSS feeds to name a few.

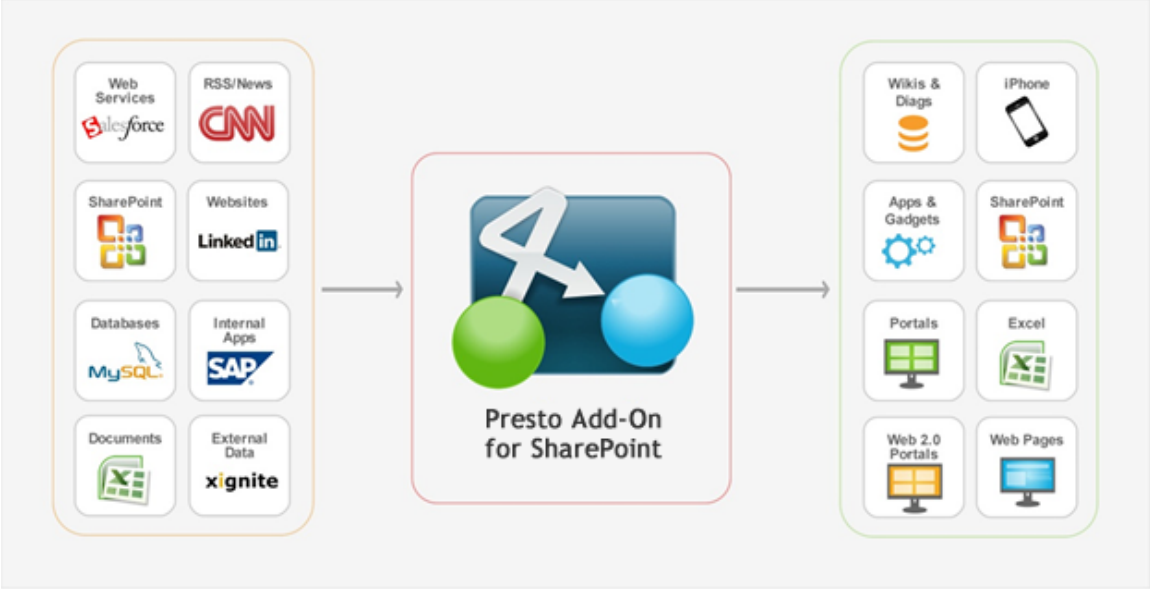
Maximize the Potential of Information

By enabling information mashing in conjunction with SharePoint's content management services, P4S delivers any data from any application to any user while meeting the toughest security and governance requirements. P4S also provides enterprise-ready security and governance, including support for seamlessly mashing NTLM-based services and single sign-on for your mashups.

Through easy-to-use visual tools that let you create, use, and share MashZone NextGen mashups and apps in SharePoint, P4S empowers users with relevant, timely information, ultimately enabling business users to make more accurate and timely decisions.

Empower SharePoint Users with Powerful Visual Tools

P4S empowers users to:



- *Create from SharePointInformation*: user-friendly, point-and click-tools let users build mashups and apps from SharePoint information sources, including lists and simple or advanced search results.
- *Publish Apps or Mashups to SharePoint*: easy wizards let users publish mashups and apps as native SharePoint lists and Web Parts.

  SharePoint Designer lets users publish mashup as external SharePoint lists for dynamic integration with other information sources connected to MashZone NextGen.
- *Rely on Integrated Security*: a secure Single Sign-On ensures the safety and security of mashups and apps combining numerous information sources used in SharePoint.
- *Include Rich Interactivity*: users can quickly create communication between their apps and other Web Parts in their SharePoint sites that enables the ability to slice and filter information for a deeply interactive user experience.
- *Use Your Current Installation*: P4S is compatible with all versions of SharePoint 2007 and SharePoint 2010.

Maximize SharePoint ROI

Rather than relying on custom development with its inherent costs and project delays, P4S maximizes the ROI of your SharePoint investment by enabling you to get SharePoint lists from sites across your SharePoint environment and mash them with other, non-SharePoint data creating a synergy of meaningful and valuable information gathered in one easy-to-access, easy-to-view place.

P4S allows organizations to unlock the value of information held in SharePoint by allowing users to rapidly combine their SharePoint data with information from large enterprise systems like ERP, SCM, CRM, and Business Intelligence.

Furthermore, P4S enables organizations to quickly and easily publish mashups and apps as native SharePoint Web Parts or as native or external lists and share them with other users.

P4S enables:

- *Improved Decisions*: access and combine any data to quickly reveal insightful contextual information.
- *Better Agility*: allows IT to offer business units with greater self-service solutions while maintaining governance and compliance to company access policies.
- *Increased SharePointROI*: bridge disparate IT environments with MashZone NextGen mashups so SharePoint users can get to the information that they need from one place.

MashZone NextGen Add-On for SharePoint in Action

Example uses of P4S include:

- *Sales Support*: commercial enterprises supporting sales reps by dynamically combining sales leads information from a CRM system with prospect profile information gathered from the web in minutes and then deliver new insight to a sales rep via the portal as a custom Web Part.
- *Disaster Planning and Response*: Insurance companies mashing up policy information with weather forecast data, and display all of the elements on Bing maps to assist in natural disaster response.

MashZone NextGen Add-On for SharePoint Features

P4S allows users to easily combine information sources from inside and outside of SharePoint. P4S enables users to publish both mashups and apps in SharePoint as native or external lists or as Web Parts, thus enabling quick and easy creation and use of mashups and apps within SharePoint sites. The following features are supported for publishing to SharePoint:

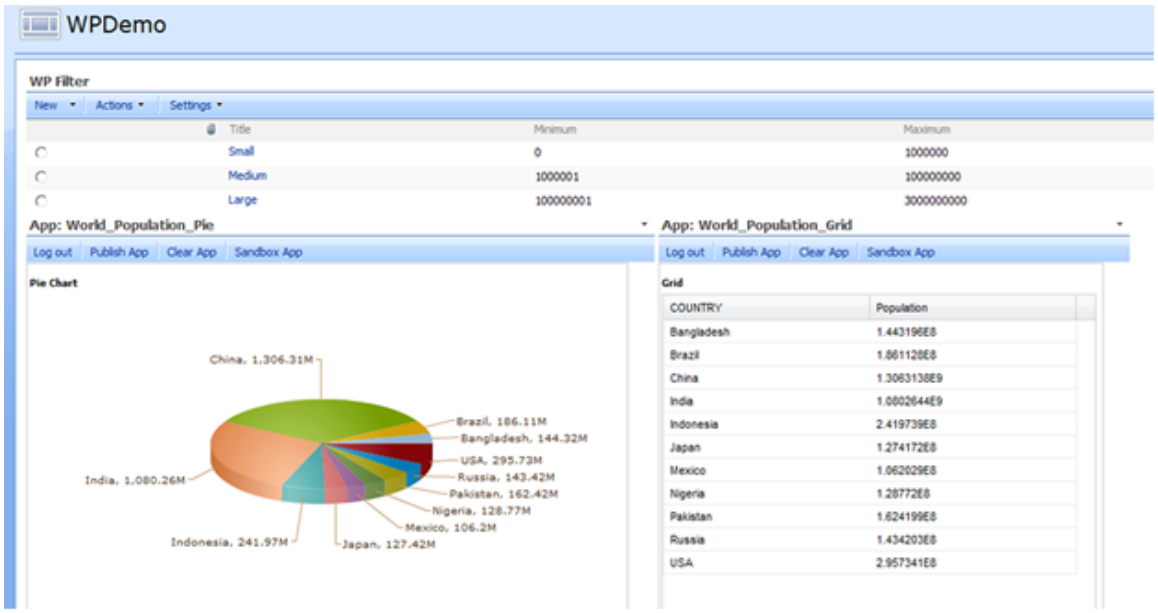
- Mashup Web Parts
- App Web Parts
- Inter-Web Part Communication
- Snapshots of mashup results as native SharePoint lists.
- Dynamic mashups results as external SharePoint lists.

P4S also enables easy use of SharePoint lists as mashable information sources to be used in mashups directly. These features include:

- Visual mashing of SharePoint lists or list views
- Visual mashing of SharePoint simple or advanced search results
- Adding items to SharePoint lists from MashZone NextGen mashups

- Easily adapting MashZone NextGen mashups for integration through BCS and external SharePoint lists

Let's have a first-hand look at some of the features of P4S and how it enhances SharePoint.



This page displays a dynamic dashboard of real-time, customizable data. The dashboard contains two apps on world population data from MashZone NextGen and a custom List Web Part from SharePoint.

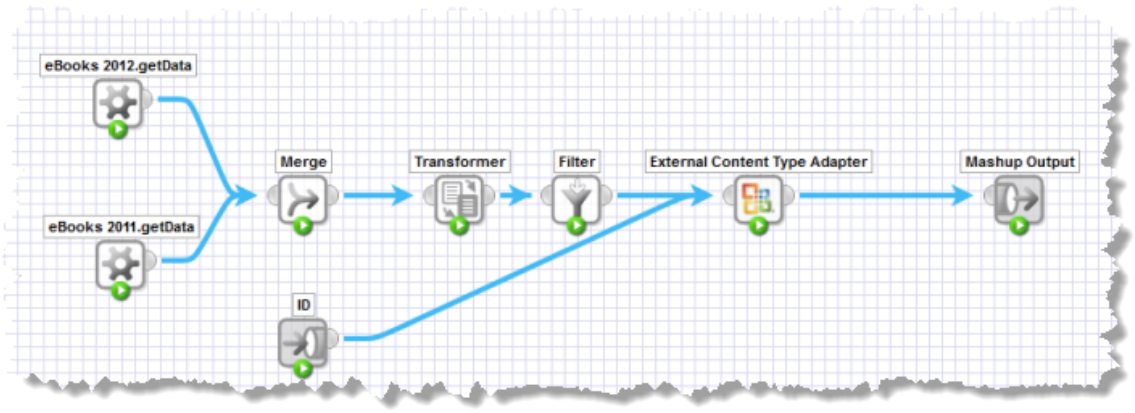
The apps filter the countries shown based on a population range. They are connected using Inter-Web Part Communication to the custom List Web Part to allow users to easily select the range of populations they want to see.

This page was created using P4S features that are integrated into SharePoint when P4S is installed along with MashZone NextGen features that are enabled with the P4S add-on. Here's a brief walk-through of the steps taken to create this page:

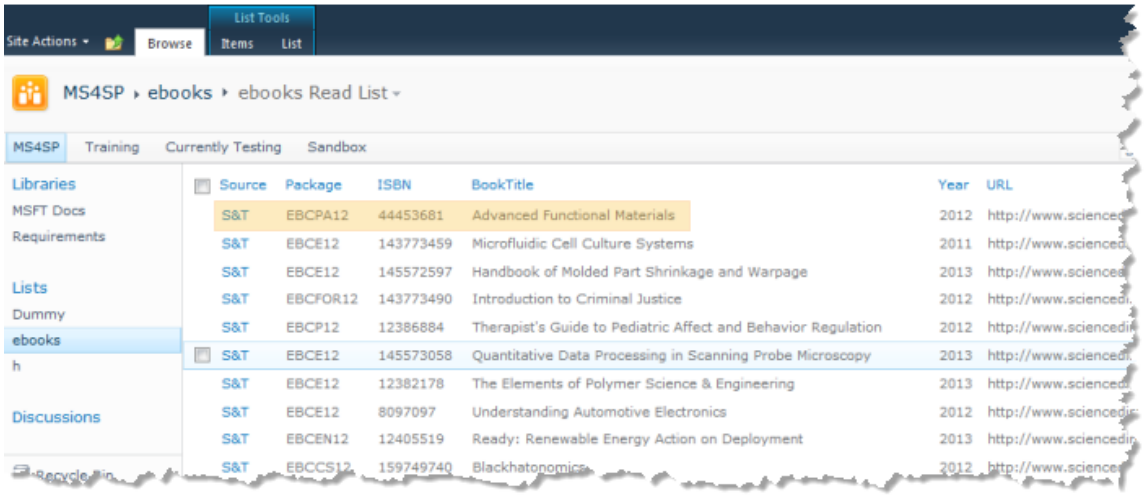
1. It starts with creating a SharePoint list containing world population data from an Excel spreadsheet.
2. This SharePoint list is registered as a mashable with MashZone NextGen and used in a mashup created in Wires in the MashZone NextGen Hub. The mashup adds filtering to select countries within a range of population.
3. The mashup has two views defined: a pie view and a grid view of the data.These views are used to create two apps in MashZone NextGen Hub. The mashup and apps are all hosted in the MashZone NextGen Server.
4. These two apps are published to SharePoint as Web Parts in this Web Part page.
5. The third Web Part contains a SharePoint Custom List (*WP Filter*) which provides the values for filtering the data shown in the two apps.

6. Both apps are connected to the Custom List Web Part using Inter-Web Part Communication. As users select a population range, both apps update the data they display.

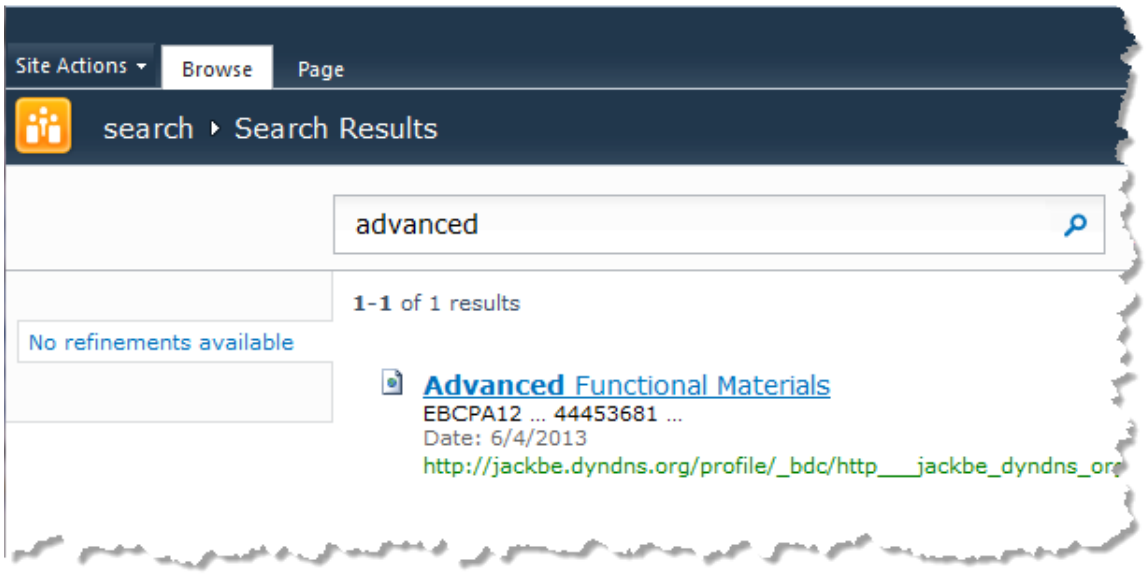
The next example shows a mashup in MashZone NextGen that merges and transforms information on eBooks from other mashables. The mashup uses the ECT Adapter block to adapt the mashup for use as external content in SharePoint, allowing dynamic updates to show in SharePoint.



Users define an external content type in SharePoint Designer based on this mashup and then add mashup results to a SharePoint site as an external List:



As an external List, the mashup results can now be searched or used in other SharePoint services:



Getting Started

You, or your SharePoint administrator, must first download and install P4S in the appropriate SharePoint farms and servers. You must also configure the P4S solution including:

- Activating P4S features.
- Determining an authentication architecture for P4S.
- Installing MashZone NextGen and deploying MashZone NextGen Servers.
- Configuring authentication, connections to MashZone NextGen Servers and other P4S properties.
- Optionally, creating a Mashup Site for developers and users to begin working with mashups and apps.

SharePoint administrators should see these guides for information on these tasks:

- "Installing the MashZone NextGen Add-On for SharePoint Solution" on page 1367
- *For 2007*: "P4S Configuration and Administration, SharePoint 2007" on page 1392.
- *For 2010*: "P4S Configuration and Administration, SharePoint 2010" on page 1372.

Once P4S is installed and configured, developers and users can begin creating mashups and apps and adding them to SharePoint. Users can:

- Create Mashup Sites if desired.
- Register Lists as mashable information sources in the MashZone NextGen Server.
- Create mashups in MashZone NextGen Hub that use SharePoint information or information from other services, feeds and sources.

- Create apps in MashZone NextGen Hub using mashups or mashables and different views, or formats of this information. Or create workspaces that combine several apps to work together.
- Add mashups directly to SharePoint Web Part pages.
- Or add mashups to SharePoint as external or native lists.
- Add apps directly to SharePoint Web Part pages.
- Use Inter-Web Part Communication in SharePoint to allow mashups and apps to work together with other SharePoint lists.

SharePoint developers and users should see the "MashZone NextGen Add-On for SharePoint User Guide" on page 1414 for information on these tasks.

For information on creating mashups and apps in MashZone NextGen Hub, developers and users should see:

- "Mashups in MashZone NextGen Wires" on page 281
- "Mashups in EMML" on page 441
- "Apps and Workspaces" on page 1091
- "Custom Apps" on page 1191

Installing MashZone NextGen Add-On for SharePoint

The MashZone NextGen Add-On for SharePoint (P4S) installs an add-on solution to your SharePoint servers that works with one or more MashZone NextGen Servers installed with MashZone NextGen.

To install the MashZone NextGen Add-On for SharePoint:

1. Download the MashZone NextGen Add-On for SharePoint installation package from Empower, the Software AG support portal, at http://empower.softwareag.com/.

   To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com.

2. Follow the installation instructions in "Installing the MashZone NextGen Add-On for SharePoint Solution" on page 1367 to install this add-on in your SharePoint environments. For upgrades see "Upgrading to New Releases of P4SP" on page 1371.
3. Download and install a compatible version of MashZone NextGen. See "Install or Upgrade MashZone NextGen Servers for P4S" on page 1372 for instructions.

Next Steps

You must add configuration to SharePoint and to MashZone NextGen to enable communication and various features in MashZone NextGen Add-On for SharePoint and the MashZone NextGen Hub. See these topics for more information:

-
- Configuring "Configure MashZone NextGen Connections to SharePoint" on page 1655
- For SharePoint 2010:
  - "Planning Authentication for P4S 2010" on page 1372
  - "Configuring MashZone NextGen Add-On for SharePoint 2010" on page 1378
- For SharePoint 2007:
  - "Planning Authentication for P4S 2007" on page 1392
  - "Configuring MashZone NextGen Add-On for SharePoint 2007" on page 1396

Installing the MashZone NextGen Add-On for SharePoint Solution

Before Your Start

Before you install the MashZone NextGen Add-On for SharePoint (P4S) solution, you must configure the following in your SharePoint environment:

- A compatible SharePoint Server shown below with optional services:

Microsoft Office SharePoint Server (MOSS) 2007	Optionally with MOSS SSO to provide single sign-on capabilities and greatly enhance the user experience. P4S single sign-on is only available with MOSS 2007 SSO enabled. MOSS 2007 also provides additional Web Parts for Inter-Web Part communication including mashup and App Web Parts from MashZone NextGen Add-On for SharePoint.
Microsoft SharePoint Server (MSS) 2010	Optionally with the Secure Store Service (SSS) to provide single sign-on capabilities and greatly enhance the user experience. Single sign-on is only available with MSS 2010 and SSS enabled. MSS 2010 also provides additional Web Parts for Inter-Web Part communication including mashup and App Web Parts from MashZone NextGen Add-On for SharePoint.

- Microsoft .NET Framework 3.5.

Although not required for general use of P4S, publishing apps from a MashZone NextGen Server to SharePoint and registering SharePoint Lists as mashables in a MashZone NextGen Server both require either NTLM v2 or Basic authentication.

If you are updating an existing deployment of P4S with a new release, see "Upgrading to New Releases of P4SP" on page 1371 for instructions.

Install the P4S Solution

The installation wizard for P4S performs two major tasks:

- Unpack and register P4S with Windows. You must use the automated wizard to perform this task.
- Deploy the P4S solution to the selected web applications or site collections in your SharePoint farm.

  You can choose to deploy P4S solution manually, if needed.

Note: You must be logged directly into the SharePoint server using a SharePoint administrator user account for installation. See "Troubleshooting Installation for the P4S Solution" on page 1369 for additional information.

1. Download the MashZone NextGen Add-On for SharePoint installation package for SharePoint 2007 or SharePoint 2010 from Empower, the Software AG support portal, at http://empower.softwareag.com/.

   To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com.
2. Log into the SharePoint server using a SharePoint administrator user account, if you have not already logged in.
3. Double-click the appropriate MSI file shown below to start the installation wizard:

SharePoint 2007	PrestoAddOnForSharePoint2007(v3.7).msi
SharePoint 2010	PrestoAddOnForSharePoint2010(v3.7).msi

   The installation wizard welcome screen opens. Click **Next**.
4. If needed, change the default folder where the installer will unpack P4S deployment files. Click **Next**.
5. Click *Next* again to confirm the installation. The wizard unpacks P4S.

   Important: If you prefer to manually deploy P4S solution in your SharePoint farm or you encounter errors during automated deployment, you can close the installation wizard at this point.Follow the manual deployment instructions in the readme.txt file in the folder where you unpacked P4S.
6. Click *Next* in the Deployment Confirmation screen to automatically deploy the P4S solution.

The deployment wizard performs a set of system checks to ensure that all deployment requirements are met before proceeding. Click *Next* once these system checks are completed to continue with the deployment.

See "Troubleshooting Installation for the P4S Solution" on page 1369 for suggestions to resolve common problems discovered in these system checks.

7. The End-User License Agreement displays. Read the license agreement and accept the terms. Click *Next*.
8. Choose the web applications or site collections for this SharePoint farm where you wish to deploy P4S. Click Next.

   Deployment begins.

   Note: If you create a new site collection after P4S has been deployed and want to use the MashZone NextGen Web Parts in this new site collection, you may activate P4S features for the new site. See "Activate P4S Features for SharePoint 2007" on page 1397 or "Activate P4S 2010 Features" on page 1378 for instructions.

9. Click **Close** once deployment is complete
10. Install MashZone NextGen Servers as needed. See "Install or Upgrade MashZone NextGen Servers for P4S" on page 1372 for instructions.

What is Installed

Installing MashZone NextGen Add-On for SharePoint solution deploys the following in SharePoint:

- WSP Package, which contains the features and content needed to use MashZone NextGen Add-On for SharePoint.
- Updates to the web.config file of the target web application to give permission for use of the of the DLLs deployed.
- Updates to the web.config file enabling session state and AJAX extensions.

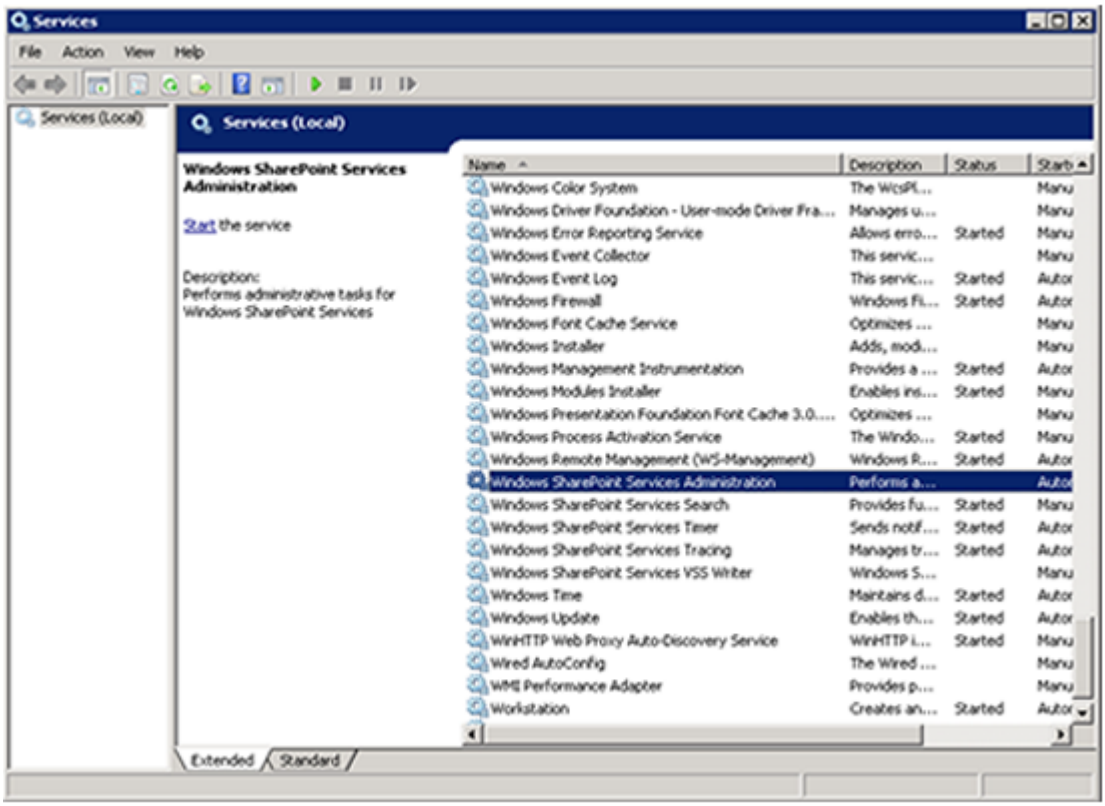
Troubleshooting Installation for the P4S Solution

System Check: Administration Service Not Started

This error occurs during system checks performed before the MashZone NextGen Add-On for SharePoint (P4S) solution is deployed. Please ensure you are logged in to the SharePoint server with an administration account and execute the following steps.

1. Click the **Abort** button to stop the current installation.
2. Navigate to Services (For Microsoft Windows Server 2008, click **Start > Administrative Tools > Services**).
3. Click the **Name** column to sort the list of services.

4. Find the service named `Windows SharePoint Services Administration`.



5. Right-click `Windows SharePoint Services Administration` and click **Start**.
6. Restart the P4S installation wizard. See "Installing the MashZone NextGen Add-On for SharePoint Solution" on page 1367 or "Upgrading to New Releases of P4SP" on page 1371.

System Check: You do NOT have permission to install and deploy solutions!

This error occurs during systems checks performed before deploying the P4S solution. This appears to happen only on systems running Microsoft Windows Server 2008. If you encounter this problem, click **Abort** to stop the deployment. Log in as a SharePoint administrator and restart the installation wizard. Right-click the MashupSitesForSharePoint(v3.5).msi file and choose **Run as Administrator**.

If deployment continues to fail, a SharePoint administrator must manually deploy the P4S solution. Follow the manual deployment instructions in the readme.txt file in the folder where you unpacked P4S.

Upgrading to New Releases of P4SP

The installation wizard for the MashZone NextGen Add-On for SharePoint solution also handles installing and deploying new releases when you have an existing version installed.

Note: You must be logged directly into the SharePoint server using a SharePoint administrator user account for installation of new releases. See "System Check: You do NOT have permission to install and deploy solutions!" on page 1370 for additional information.

1. Download the MashZone NextGen Add-On for SharePoint installation package from Empower, the Software AG support portal, at http://empower.softwareag.com/.

   To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com.

2. Log into the SharePoint server using a SharePoint administrator user account, if you have not already logged in.
3. Double-click the appropriate MSI file to start the installation wizard:

SharePoint 2007	PrestoAddOnForSharePoint2007(v3.7).msi
SharePoint 2010	PrestoAddOnForSharePoint2010(v3.7).msi

4. Leave the **Repair** option selected and click **Next**.

   If you choose to remove the existing solution, instead, configuration for this solution will be removed. Once the new release is installed, you must configure this solution.

5. Once the installation step is complete, click **Close**.
6. Click **Next** to begin deployment of the solution for this release.

   The installation wizard completes several checks to ensure that deployment is possible. See "Troubleshooting Installation for the P4S Solution" on page 1369 for information on common errors.

7. Once all checks are successful, click **Next**
8. Leave the **Repair** option selected and click **Next**.

   Deployment begins.

9. Once deployment is successful, click **Next**
10. Click **Close**.

11. Upgrade the MashZone NextGen Servers in your environment to the compatible version MashZone NextGen and apply fixes, if any are available. See "Install or Upgrade MashZone NextGen Servers for P4S" on page 1372 for instructions.

Install or Upgrade MashZone NextGen Servers for P4S

To install new MashZone NextGen Servers or upgrade existing servers for your P4S solution

1. Download the compatible version of MashZone NextGen from Empower, the Software AG support portal, at http://empower.softwareag.com/.

   See "Additional MashZone NextGen System and Software Requirements" on page 1564 for information on compatibility.

   To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com.

2. Install the MashZone NextGen Servers as needed. See *Installing Software AG Products* for instructions.

3. Apply the latest fixes, if any, to MashZone NextGen using the Software AG Update Manager. See *Using the Software AG Update Manager* for instructions.

P4S Configuration and Administration, SharePoint 2010

Once you have installed the MashZone NextGen Add-On for SharePoint (P4S) solution, configuration involves activating features across your farms and sites, setting up authentication mechanisms between SharePoint and MashZone NextGenMashZone NextGen Servers and defining connections to the MashZone NextGen Servers. See "Planning Authentication for P4S 2010" on page 1372 and "Configuring MashZone NextGen Add-On for SharePoint 2010" on page 1378 for details.

For administration tasks, see "P4S Farm and Site Administration, SharePoint 2010" on page 1392.

Planning Authentication for P4S 2010

There are several options you can use to implement and optimize authentication between your SharePoint installation and MashZone NextGenMashZone NextGen Servers. This allows you to set up an appropriate level of security and user experience when your environment has bi-directional authentication between SharePoint and one or more MashZone NextGen Servers.

The challenge for user authentication with MashZone NextGen Add-On for SharePoint (P4S) 2010 is that users, who are already authenticated in SharePoint, must also authenticate with one or more MashZone NextGen Servers as seamlessly as possible. For the best user experience, a user should only be required to login to a system once and gain authorization to remote systems using their current identity.

To define the appropriate plan for authentication, you must consider the various User Authentication Experiences for P4S 2010 and Authentication Designs for P4S 2010 based on your SharePoint environment.

User Authentication Experiences for P4S 2010

There are three potential authentication experiences for users working with Web Part pages that contain mashups or apps using MashZone NextGen Add-On for SharePoint (P4S) 2010. From best experience to worst, these are:

- *Full Single Sign-On*: with this experience, a user logs into SharePoint. Their SharePoint credentials are used without any additional login challenges to authenticate with MashZone NextGen Servers for any mashups or apps the user may work with in Web Part pages.
- *Initial Challenge Only*: this provides a partial SSO experience limiting login challenges to a single challenge for the first mashup or app that a user encounters in a Web Part page. In addition to logging in with SharePoint, a user must login with MashZone NextGen Servers once during a session in SharePoint (a browser session). Each new session in SharePoint may require this initial login to MashZone NextGen.
- *All Challenges:* no authentication information is shared between SharePoint and MashZone NextGen with this design. Users must login with MashZone NextGen Servers for *every* mashup or app that requires authentication. Obviously, this is the least user-friendly experience.

You choose an appropriate user experience based the Authentication Designs for P4S 2010 available for your SharePoint environment.

Authentication Designs for P4S 2010

The user experiences that you can deliver using MashZone NextGen Add-On for SharePoint (P4S) 2010 depend on your SharePoint environment and how SharePoint and MashZone NextGen Servers are deployed. The factors that determine an authentication architecture include:

- *SharePointVersion and Authentication*: using Microsoft SharePoint Server (MSS) 2010:
  - *With the Secure Store Service* to provide a true single sign-on experience for P4S users. Other factors in your environment also play a part in supporting single sign-on.

    The Secure Store Service stores user credentials for additional applications, such as the MashZone NextGenMashZone NextGen Server, accessed from SharePoint. User credentials are organized by *target application*, also known as an *SSO Application Name*, allowing the Secure Store service to handle credentials for different applications. You can store user credentials under one SSO Application name to provide credentials that several MashZone NextGen Servers share, allowing users to access mashups and apps from those MashZone NextGen Servers seamlessly, or keep credentials for some MashZone NextGen Servers separate using different SSO Application Names.

- *WithoutSecure Store Service:* this prevents a full single sign-on experience, but can still support SSO with an initial challenge experience depending on other factors in your SharePoint environment.

> **Note:** P4S is not currently compatible with third-party single sign-on solutions, such as Netegrity SiteMiner, that are agent-based. If your SharePoint environment uses a third-party SSO solution, users may work in MashZone NextGen Hub with SharePoint to create mashables, mashups and apps. However, users cannot use P4S actions in SharePoint and cannot publish mashups or apps to SharePoint.

- *SharePointand MashZone NextGenMashZone NextGen ServerDomains:* how you deploy MashZone NextGen Servers in your SharePoint environments affects the choices for authentication. If single sign-on is not possible, deploying SharePoint servers and MashZone NextGen Servers in the same domains enables the use of cookies to simplify authentication.
- *User Repository:* sharing the same user repository, such as an Active Directory or LDAP Directory, for both SharePoint servers and MashZone NextGenMashZone NextGen Servers supports a true single sign-on experience. This enables MashZone NextGen Servers to use the same credentials as SharePoint.

> **Note:** Currently there is a known issue for authentication challenges for mashups published in Mashup Web Parts when the user repository is shared. For mashups users will be challenged initially for mashups.

The combination of these factors produce three authentication architectures that you can use. The following table summarizes which architectures can be applied to specific environments and the resulting user experience:

Authentication Architecture	MSS + Secure Store	MSS + No Secure Store	SharePoint / MashZone NextGen Server in Same Domains	SharePoint / MashZone NextGen Server in Different Domains	SharePoint / MashZone NextGen Server Share User Repository	User Experience
SSO + Token Authentication	✓		✓	✓	✓	Full SSO
	✓		✓	✓		Initial challenge
SSO + Cookie Authentication		✓	✓			Initial challenge

Authentication Architecture	MSS + Secure Store	MSS + No Secure Store	SharePoint / MashZone NextGen Server in Same Domains	SharePoint / MashZone NextGen Server in Different Domains	SharePoint / MashZone NextGen Server Share User Repository	User Experience
			✓			Initial challenge
No SSO, Manual Authentication		✓		✓		All challenges
				✓		All challenges

SSO + Token Authentication

This authentication design can be used when both criteria are true:

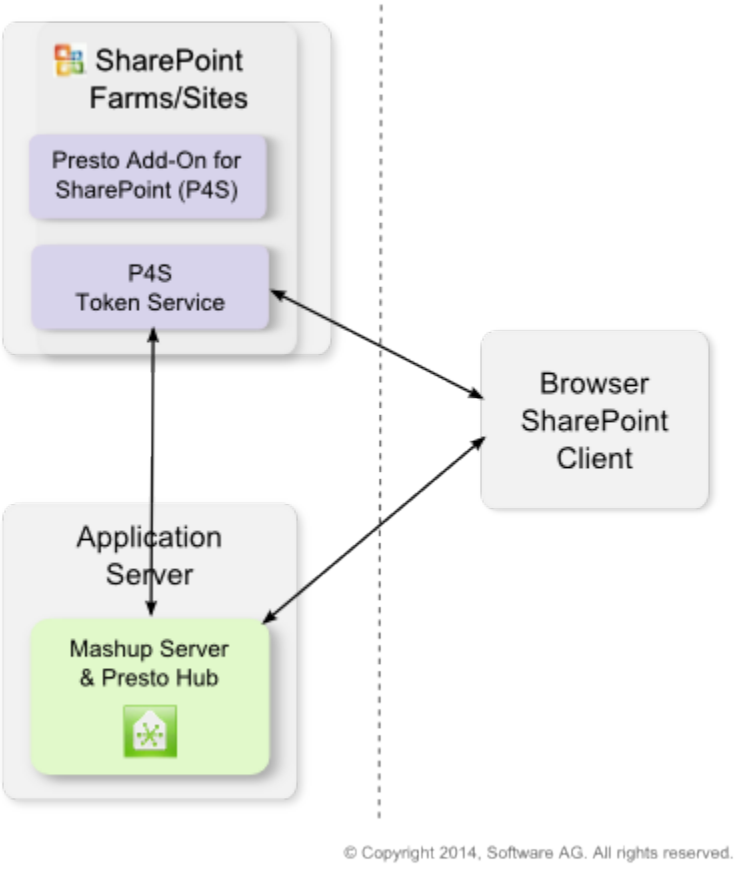
- You have MSS 2010 in your environment
- You have chosen to use the Secure Store service

This solution uses the Token Service that is installed with P4S. The Token Service generates a token ID for a user's SharePoint session when they access mashups or apps. The requests sent to the MashZone NextGen Servers that host these mashups or apps include:

- A ticket with the user's token ID
- An SSO Token Server Name.

  MashZone NextGen Servers use this name to find connection information to the Token Service. This name ensures that connection information to the Token Service is not passed directly in requests.

Both SharePoint and MashZone NextGen Servers must be configured with the SSO Token Server Name and the connection information to communicate with the Token Service. MashZone NextGen Servers use this SSO Token Server Name to connect to the Token Service and redeem the user ticket and credentials.



The actual user experience with an SSO + Token solution depends on whether the MashZone NextGen Servers and SharePoint servers share an Active Directory as their user repository.

With a Shared User Repository

If SharePoint and the MashZone NextGen Servers for your mashup sites also share the same user repository, such as an Active Directory or LDAP Directory, this results in a full single sign-on experience for users. Users login to SharePoint and no further login challenges are issued when they access mashups or apps in MashZone NextGen Servers in this domain. MashZone NextGen Servers retrieve user credentials via tokens and then authenticate these credentials and retrieve user authorization information against the shared user repository.

Note: MashZone NextGen Servers use only basic user credentials (username and password). They do not accept NTLM credentials which include Windows domains as part of the user name.

With Distinct User Repositories

If SharePoint and the MashZone NextGen Servers in this domain do not share a user repository, users receive one login challenge the first time they access mashups or apps

hosted in a MashZone NextGen Server for a given SSO Application Name (a target application) in SharePoint configuration.

The credentials they enter for this initial login challenge are then stored by the Secure Store service under the SSO Application configured for that connection. For all subsequent requests to MashZone NextGen Servers with that same SSO Application name, MashZone NextGen Servers retrieve user credentials via tokens and the Token Service and then authenticate them and retrieve user authorization information from the MashZone NextGen User Repository.

You can have each SSO Application store user credentials for one or several MashZone NextGen Server connections in SharePoint.

SSO + Cookie Authentication

This authentication design can be used when you have MSS 2010 in your environment, but have chosen *not* to use the Secure Store service *and* the SharePoint servers and MashZone NextGen Servers for your mashup sites are located in the same domain.

Same Domains

Domains are the same for SharePoint and MashZone NextGen Servers when the domain names fit either of these patterns:

`*.`*domain-name.type*

domain-name.type `:`*different-port*

For example, if the SharePoint server domain is `site1.com`, then a MashZone NextGen Server in the domain `ms.site1.com` or `site.com:8080` would be in the same domain.

> **Note:** The patterns for matching domain names are a restriction of Internet browsers.

With cookie forwarding, a user must login once with a MashZone NextGen Server. These credentials are stored as a cookie in the browser where the user is working in SharePoint. This cookie is passed by the browser to the MashZone NextGen Server in subsequent requests so no further login challenges are issued.

> **Note:** MashZone NextGen Servers use only basic user credentials (username and password). They do not accept NTLM credentials which include Windows domains as part of the user name.

No SSO, Manual Authentication

This design is required when you have chosen *not* to use the Secure Store service *and* some MashZone NextGen Servers reside in different domains than your SharePoint servers.

Because user credentials are not stored in the Secure Store repository, MashZone NextGen Servers cannot retrieve these credentials via the Token Service. Because the SharePoint and MashZone NextGen Servers are in different domains, cookie forwarding cannot bridge this gap.

This results in manual authentication, requiring users to login to each MashZone NextGen Server as they access mashups or apps hosted by that MashZone NextGen Server to create a session or when an existing session with a MashZone NextGen Server expires.

Configuring MashZone NextGen Add-On for SharePoint 2010

You should first determine the authentication mechanisms you will need based on your SharePoint environments and the MashZone NextGen Servers that SharePoint will connect to. See "Planning Authentication for P4S 2010" on page 1372 for the concepts and options you need to determine an appropriate authentication plan.

Once you know what authentication mechanisms you need, configuration for Mashup Sites for SharePoint includes:

- Activate P4S 2010 Features: you must activate features before continuing with other configuration steps.
- Configure Authentication for P4S 2010: this is required if you are using MSS and the Secure Store Service.
- Configure MashZone NextGen Server Connections in SharePoint 2010.
- Complete Miscellaneous Configuration for P4S 2010.

Activate P4S 2010 Features

MashZone NextGen Add-On for SharePoint (P4S) adds specific actions to SharePoint farms and sites. It also defines a *Mashup Site* template which you or users can use and two new Web Parts for mashups and apps respectively.

To begin configuration, you must first enable P4S features for each SharePoint farm and site collection where users should be allowed to work with mashups and apps. See "Enabling P4S Features in Farms" on page 1378 and "Enabling P4S Web Parts for Site Collections" on page 1379.

You can also choose to manually enable P4S features for individual sites that are not Mashup Sites. See "Manually Enabling P4S Features for a Site" on page 1379.

Enabling P4S Features in Farms

First, enable Mashup Site features for the appropriate SharePoint farms:

1. Open SharePoin Central Administration as a farm administrator.
2. Click **System Settings** and click **Manage Farm Features** in the Farm Management category.
3. Find the MashZone NextGen Custom Farm Site Actions item and click **Activate** to activate these features or **Deactivate** to deactivate them.

Activating this feature adds a new category, MashZone NextGen Administration, in the Manage Applications section of Central Administration.

Enabling P4S Web Parts for Site Collections

Next, enable the new Web Parts for P4S in every site collection where users should be allowed to work with mashups or apps:

1. Log in as the administrator for a site collection. If needed, navigate to the parent site for the entire collection.
2. Click **Site Actions > Site Settings**.
3. Click **Site Collection Features** in the Site Collection Administration section.
4. Find MS4SP Web Parts and click **Activate** to activate these features or **Deactivate** to deactivate them.

This adds the Mashup Web Part and the App Web Part as valid Web Parts that users can add to Web Part pages. These Web Parts allow users to publish mashups and apps to SharePoint.

Manually Enabling P4S Features for a Site

When you create new sites using the Mashup Site template, P4S Features are automatically enabled. If you want to add these features to another type of site:

1. Login is as site administrator for the specific site.
2. Click **Site Actions > Site Settings**.
3. Click **Manage Site Features** in the Site Actions category.
4. Find MS4SP Custom Site Actions and click **Activate** to activate these features or **Deactivate** to deactivate them.

This adds a new category, MashZone NextGen Administration, in Site Settings with functions that allow you to define connections to MashZone NextGen Servers and to set other P4S configuration properties (see "Configure MashZone NextGen Server Connections in SharePoint 2010" on page 1385 for a complete list). This also adds the Register List action to allow users to register SharePoint Lists as mashable information sources in MashZone NextGen.

Configure Authentication for P4S 2010

The tasks you need to complete to configure authentication between MashZone NextGen Add-On for SharePoint (P4S) and MashZone NextGen Servers depends on the authentication plan you need to implement:

- *For SSO + Token Authentication*, you use the Secure Store Service to manage user credentials for MashZone NextGen Servers. In addition, this uses tokens managed by the Token Service from P4S and authentication configuration in MashZone NextGen Server connections.

  Configuration steps include:

  1. Configure the Secure Store Service.

2. Once the Secure Store Service is configured, you Create Secure Store Target Applications to hold user credentials for the MashZone NextGen Servers in your environment.
3. Then Create or Manage SSO Token Server Names to manage user token IDs.
4. And finally configure authentication properties for each MashZone NextGen Server connection. See "Add a MashZone NextGen Server Connection to SharePoint" on page 1386 or "Edit a MashZone NextGen Server Connection" on page 1389 for instructions.

- *For No SSO + Cookie Authentication*, you must configure cookie forwarding authentication in the properties for each MashZone NextGen Server connection. See "Add a MashZone NextGen Server Connection to SharePoint" on page 1386 or "Edit a MashZone NextGen Server Connection" on page 1389 for instructions.

  Cookie forwarding is enabled for user connecton overrides by default, so no additional configuration is required.

- *For No SSO, Manual Authentication*, you must:
  - Clear all authentication properties in each MashZone NextGen Server connection. See "Add a MashZone NextGen Server Connection to SharePoint" on page 1386 or "Edit a MashZone NextGen Server Connection" on page 1389 for instructions.
  - And Disable Cookie Forwarding for MashZone NextGen Server Overrides.

For more information on the requirements and user experience of these authentication designs, see "Authentication Designs for P4S 2010" on page 1373.

Configure the Secure Store Service

Typically, you create an instance of the Secure Store Service and configure this service when you install SharePoint using the configuration wizard. For more information on deployment and configuration for the Secure Store Service, see the Microsoft TechNet articles:

- Plan the Secure Service Store
- Configure the Secure Service Store

There are no additional requirements to configure the Secure Store Service for MashZone NextGen Add-On for SharePoint.

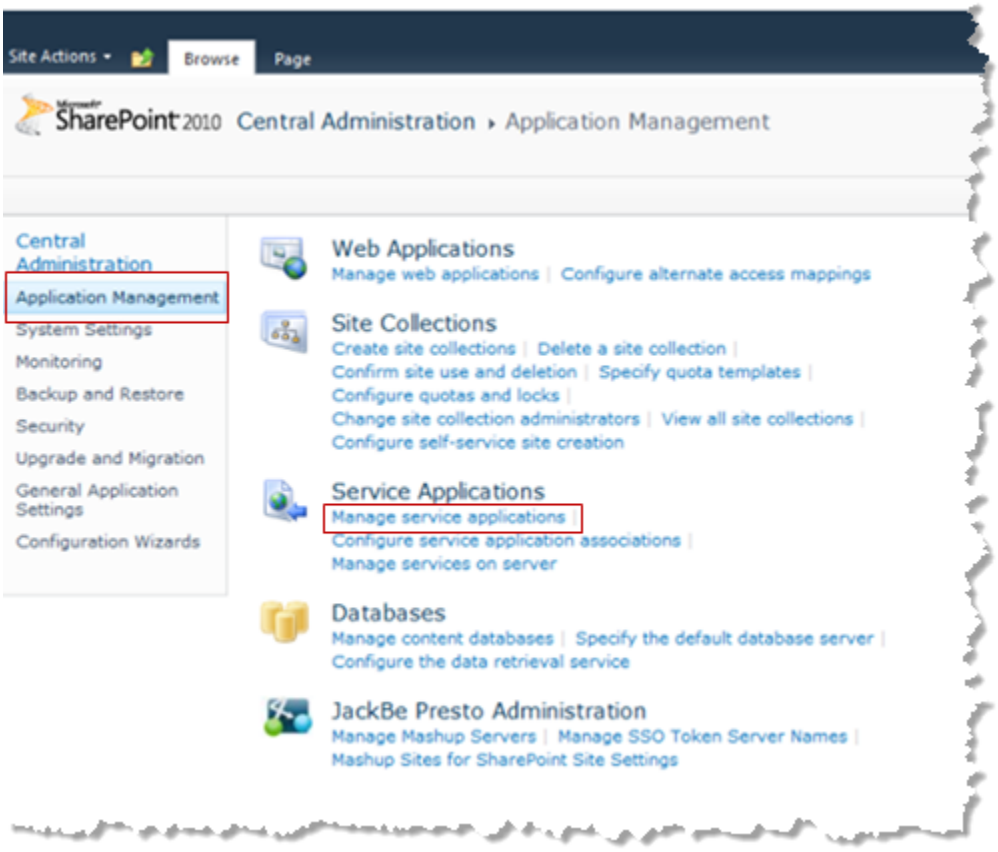
Create Secure Store Target Applications

Target applications organize user or group credentials for SharePoint users to access other systems, such as MashZone NextGenMashZone NextGen Servers. You create a target application to provide credentials when users connect to MashZone NextGen Servers to find or work with mashups and apps in Web Part pages.

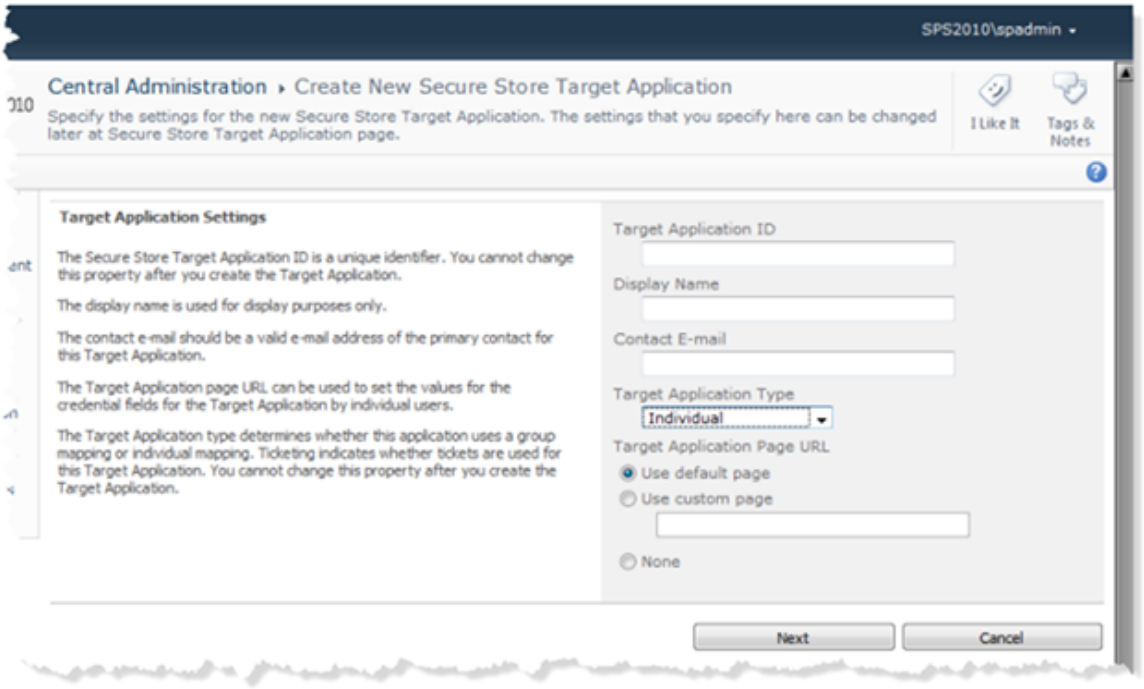
To create target applications:

1. Open SharePoint Central Administration as a farm administrator.

2. Click **Application Management** and click **Manage service applications** in the Service Applications category.



3. Select the Secure Store Service in the list and click **Manage** in the toolbar ribbon.
4. Click **New** and complete these properties:



- **Target Application ID** = a unique ID for this target application.
- **Display Name** = the name to display to users for this target application. This is the SSO Application Name that users or administrators assign to MashZone NextGen Server connections.
- **Contact E-mail** = the email address of the primary contact for MashZone NextGen Add-On for SharePoint and the MashZone NextGen Servers associated with this target application. This can be any valid email address.
- **Target Account Type** = the type of user account mapped by this target application. Typically, this is `Individual Ticket`, to allow individual user credentials to be mapped. You can also `Group Ticket` as a type to associate all users in groups to one set of credentials.
- **Target Application Page URL** = leave the Use default page option set.
- **Ticket Timeout** = if needed, change the default timeout period for tickets generated for this Target Application.

5. Click **Next**.
6. Leave the default fields that appear for user credentials. Click **Next**.
7. If needed, find and add other SharePoint administrators or users to the **Target Application Administrators** list who should be allowed to manage this target application. The farm administrator has access by default.
8. If needed, find and add other SharePoint administrators or users to the **Ticket Redeemers** list who should be able to redeem tickets for this target application.

9. Click **OK**.

Once you have a target application, you can create or update MashZone NextGen Server connections with this SSO application information. See "Configure MashZone NextGen Server Connections in SharePoint 2010" on page 1385 for instructions.

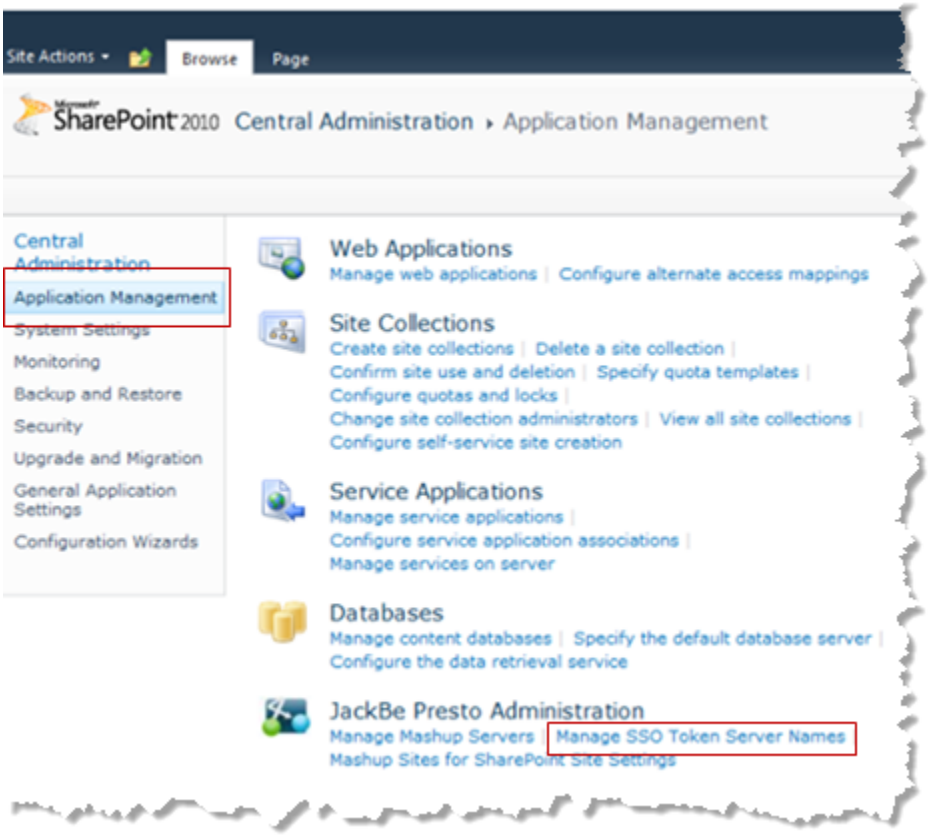
Create or Manage SSO Token Server Names

You must create an SSO Token Server name for each domain with a SharePoint farm or server that connects to MashZone NextGenMashZone NextGen Servers. SSO Token Server Names are a placeholder that is substituted in requests for the host portion, such as http://myOrg.com, of the actual URL that MashZone NextGen Servers use to connect to this Token Service. MashZone NextGen Add-On for SharePoint sends the SSO Token Server Name to MashZone NextGen Servers which use this name to interact with the Token Service to validate SSO credentials when users connect to the MashZone NextGen Server. See "SSO + Token Authentication" on page 1375 for more information.

Note: In addition to configuration in SharePoint, each MashZone NextGen Server that has a connection in SharePoint must also have configuration for the SSO Token Server Name. For more information, see "Configure MashZone NextGen Connections to SharePoint" on page 1655.

To create or manage SSO Token Server Names:

1. Open SharePoint Central Administration as a farm administrator.
2. Click **Application Management** and click **Manage SSO Token Server Names** in the JackBe MashZone NextGen Administration category.



3. To add a name, click **New Token Server Name** and complete these properties:
   - **SSO Token Server Name** = this is the placeholder value to display for this Token Service. This is also used in requests to MashZone NextGen Servers to allow them to find the correct connection information for this Token Service.
   - **SSO Token Server URL** = the URL to this Token Service.
4. Then click **OK**.

To edit or delete a name, click the down arrow for an existing SSO Token Server Name and select **Edit SSO Token Server Name** or **Delete SSO Token Server Name**.

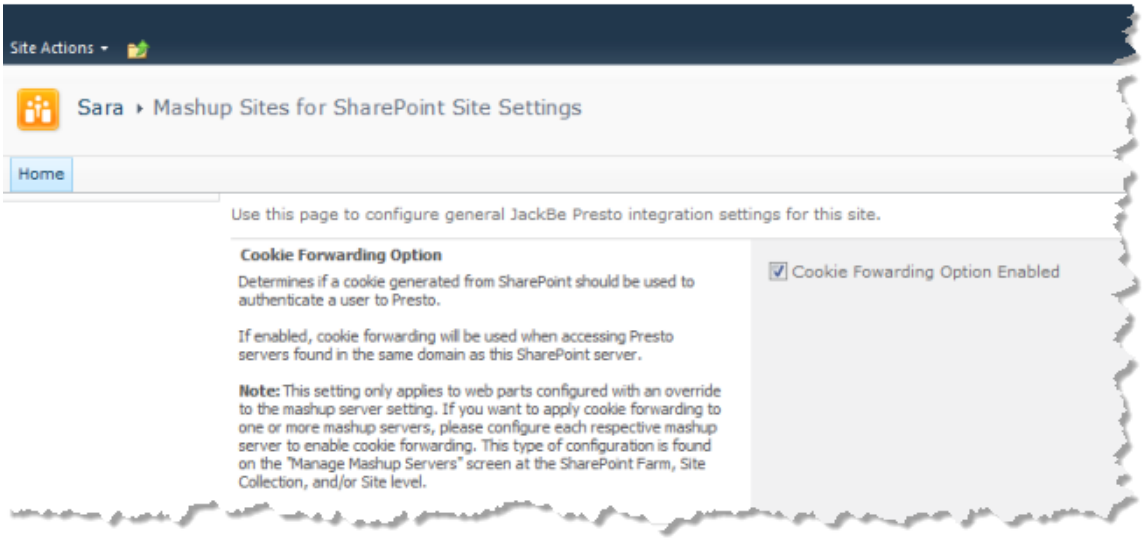
Disable Cookie Forwarding for MashZone NextGen Server Overrides

When users choose the preconfigured MashZone NextGen Server connection in Web Parts, all aspects of the connection including authentication is defined in the connection. With user overrides, cookie forwarding is enabled by default.

If you are using No SSO + Manual authentication, you should disable cookie forwarding for user overrides. You do this in site settings for a site collection or a specific site.

1. Login is as site administrator for the site collection or specific site.
2. Click **Site Actions > Site Settings**.
3. Click **Mashup Sites for SharePoint Site Settings**.

4. Clear the **Cookie Forwarding Option Enabled** option and click **OK**.



Configure MashZone NextGen Server Connections in SharePoint 2010

MashZone NextGen Add-On for SharePoint (P4S) allows you to connect multiple MashZone NextGen Servers to many SharePoint sites and site collections, based on your P4S license. After a connection is configured in both SharePoint and MashZone NextGen, users can publish mashups and apps to SharePoint as native Web Parts. Users can also use information from SharePoint as mashable information sources in MashZone NextGen for mashups or apps.

For each connection, the visibility of the mashups and apps hosted in a MashZone NextGen Server within SharePoint and the visibility of SharePoint information within a MashZone NextGen Server is based on:

- Where your define the connections in SharePoint.

  Farm administrators or site administrators can configure MashZone NextGen Server connections to farms, site collections or individual sites. You can also define "Duplicate Connections to MashZone NextGen Servers" on page 1390 if needed.

  Users who publish apps or mashups to Web Parts in SharePoint can also override the default connections for a specific Web Part. This ensures that users can control settings at a granular level if needed. For more information about overriding MashZone NextGen Server connections in Web Parts, see "Update Mashup Web Part Properties" on page 1424 and "Update App Web Part Properties" on page 1441.

- How connections to SharePoint are defined in the MashZone NextGen Server.

For a discussion of the common patterns for configuring connections between SharePoint and MashZone NextGen Servers, see "Configure MashZone NextGen Connections to SharePoint" on page 1655.

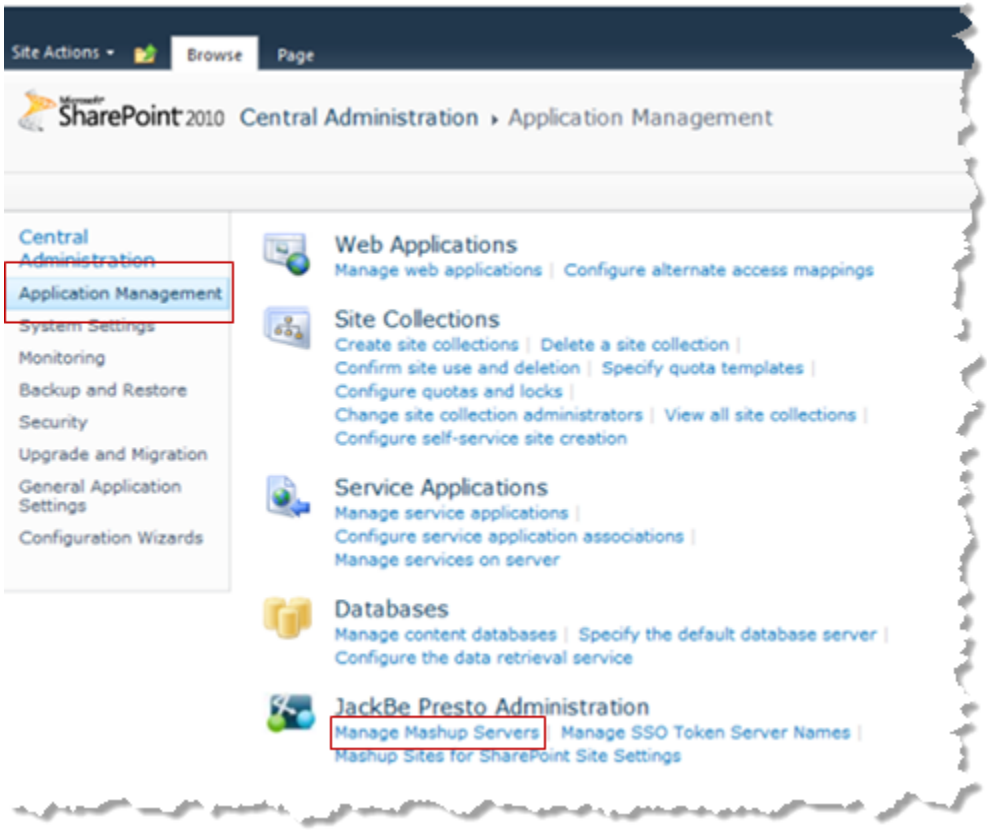
Then Add a MashZone NextGen Server Connection to SharePoint or Edit a MashZone NextGen Server Connection as needed.

Add a MashZone NextGen Server Connection to SharePoint

The scope of the connection that you create is determined by your account and the farm or site where you add the connection.

Visibility for connections can also be affected by where you add the connection. MashZone NextGen Server connections defined for site collections are immediately visible while visibility at sites and site collections for MashZone NextGen Server connections defined for farms may *be delayed* up to 30 minutes because of caching.

1. Verify or complete the following *required* prerequisites:
    a. Install the MashZone NextGen Server, if you have not already done so. See "Install or Upgrade MashZone NextGen Servers for P4S" on page 1372 for instructions.
    b. Make sure the MashZone NextGen Server is running. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
    c. Configure connections in the MashZone NextGen Server to the appropriate SharePoint farms, site collections and sites. See "Configure MashZone NextGen Connections to SharePoint" on page 1655 for instructions.
2. To create a connection to a MashZone NextGen Server in a farm:
    a. Open SharePoint Central Administration as a farm administrator.
    b. Click **Application Management** and click **Manage Mashup Servers** in the MashZone NextGen Administration category.



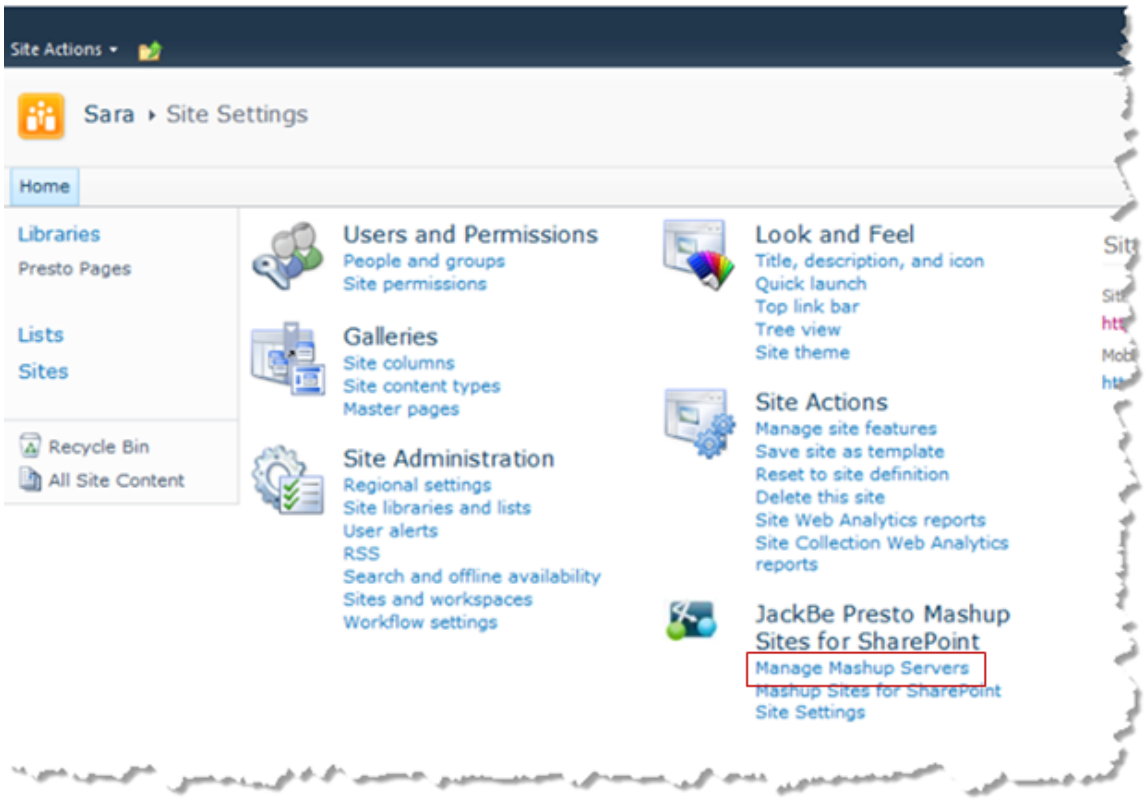
c. Skip to Step 4.

3. To create a connection to a MashZone NextGen Server in a site collection or a specific site:

   a. Login is as site administrator for the site collection or specific site. If needed navigate to the home page for the site you want.

   b. Click **Site Actions > Site Settings**.

   c. Click **Manage MashupServers** in the MashZone NextGen Administration category.



4. Click **New Mashup Server** from the Manage Mashup Server page. The Add Mashup Server page opens.
5. Complete basic properties for this connection:
   - **Name:** choose a name that describes the location or purpose of the MashZone NextGen Server you are connecting to.
   - **URL:** enter the domain name and port, or IP address, for the URL to connect to this MashZone NextGen Server. For example: `http://mashups.myorg.com:8080` or `http://204.10.1.101:8080`.

   Note: You may omit the port number in this URL if the MashZone NextGen Server uses port 80.

   URLs cannot use the `localhost` domain name. Use the IP address instead.
6. Set authentication properties for this connection based on your authentication plan. See "Authentication Designs for P4S 2010" on page 1373 for more information on the types of authentication you can use.

   If you are using SSO + Token authentication, set these properties:
   - **SSO Enabled:** set this option to enable single sign-on authentication. You must have the Microsoft SharePoint Server (MSS) 2010 and use the Secure Store Service to provide single sign-on for MashZone NextGen Server connections.

- **Shares same Active Directory as** SharePoint**:** set this option when the MashZone NextGen Server and SharePoint are sharing an Active Directory as the user repository.

  See "Authentication Designs for P4S 2010" on page 1373 for more information on the impacts of sharing a user repository and authentication.

- **SSO Application:** if you have enabled SSO (see above), choose the appropriate target application to use for authentication of credentials when connecting to this MashZone NextGen Server.

  If no SSO applications are listed, then SSO is not properly configured. See "Configure the Secure Store Service" on page 1380 for more information to correct this problem.

If you are using No SSO + Cookie authentication, set this property:

- **Cookie Forwarding:** set this option to allow cookies to handle authentication credentials between SharePoint servers in this farm and this MashZone NextGen Server. The MashZone NextGen Server*must* be in the same domain as this SharePoint farm.

If you are using No SSO, Manual authentication, clear all the SSO and Cookie properties.

7. Set the **Mashup Server Enabled** option to activate this connection and click **Verify Connection.**

   If the connection fails, check to make sure the URL you entered is correct and the MashZone NextGen Server you are connecting to is running. See "Start and Stop the MashZone NextGen Server" on page 1570 for startup instructions.

   A Confirmed message displays once your connection is verified.

8. Once you have verified the connection, click **OK**. The Manage Mashup Servers page appears, listing your new MashZone NextGen Server connection.

Edit a MashZone NextGen Server Connection

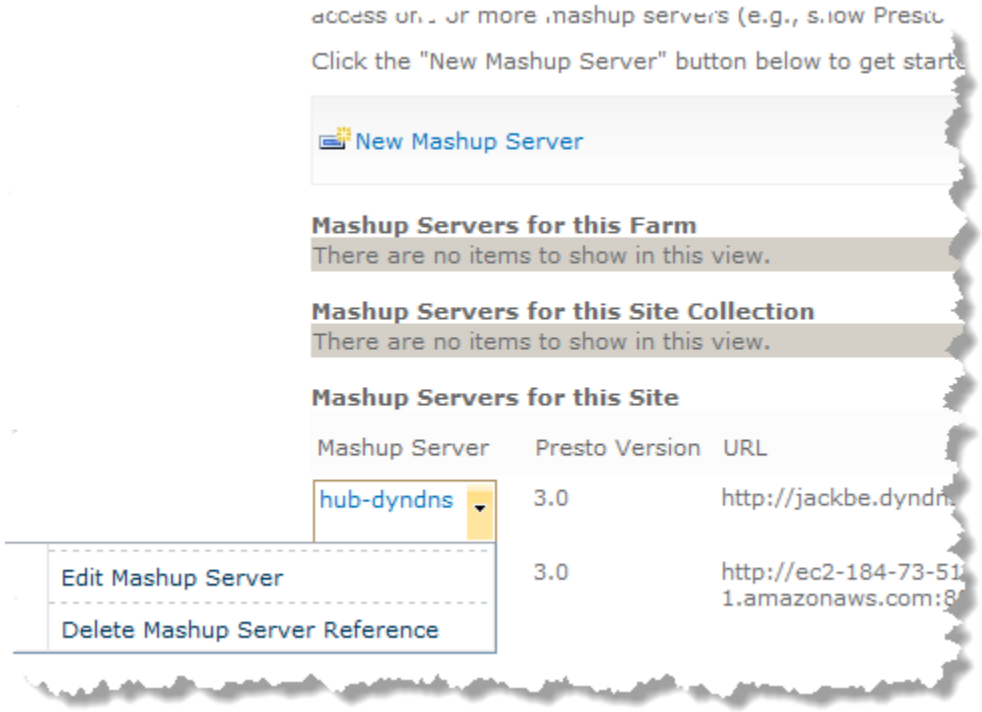
1. Find the MashZone NextGen Server Connection based on the scope of the connection:

   - For connections with a farm scope, open SharePoint Central Administration as a farm administrator. Click **Application Management** and click **Manage Mashup Servers** in the MashZone NextGen Administration category.

   - For connections with a site collection or site scope, login is as site administrator for the site collection or specific site. Click **Site Actions > Site Settings**. Click **Manage Mashup Servers** in the MashZone NextGen Administration category.

The Manage Mashup Servers page opens listing all existing connections in this scope.

2. Click the drop-down menu for the MashZone NextGen Server connection you need to edit and select **Edit Mashup Server**.



3. Update the properties you need to edit and click **Verify Connection**.
4. Once the connection is confirmed, click **OK**.

Duplicate Connections to MashZone NextGen Servers

You can define duplicate MashZone NextGen Server connections with different scopes, such as defining one connection for a SharePoint farm and a duplicate connection for a specific site collection in that farm. This can be useful, for example, if you need to configure a different authentication mechanism for a specific site collection.

When duplicate connections are defined, the site-collection-specific connection overrides the farm-level connection for that specific site collection. All other site collections inherit the farm-level connection.

Complete Miscellaneous Configuration for P4S 2010

There are two additional configuration options that you can set for farms, site collections or individiual sites that pertain to apps:

- "Set the Scope of Apps in the App Gallery" on page 1391 allows you to have finer control over which apps users can choose to add to App Web Parts from the App Gallery.

- "Set the Default Sandbox Mode for apps" on page 1391 allows you to determine whether apps are automatically isolated (in a sandbox) or not (embedded) in App Web Parts. This can affect performance, but also may affect style and JavaScript interactions with the Web Part page.

Set the Scope of Apps in the App Gallery

You have an option for SharePoint farms to determine whether 'production' apps or 'development' apps are listed in the App Gallery when users publish apps to Web Part pages.

When users create apps in the MashZone NextGen Hub, they may not be completely finished or 'production-ready' for all users to work with. Once apps are complete, users publish apps to the MashZone NextGenAppDepot where they are reviewed and approved by an AppDepot administrator. Published apps are now 'production-ready' and available for other users to work with.

By default, the App Gallery shows SharePoint users all the apps from the MashZone NextGen Hub. This allows users to build or update Web Part pages based on newly developed apps. To ensure that users only see 'production-ready' apps in a SharePoint farm, you can instead limit the App Gallery to those apps which have been pulished in the AppDepot.

To control app visibility in the App Gallery for all sites in a SharePoint farm

1. Login as a farm administrator and navigate to Central Administration.
2. Click **Application Management**, then click **Mashup Sites for SharePoint Site Settings** in the MashZone NextGen Administration category.
3. Set either the **List Apps from** MashZone NextGen Hub or **List Apps from AppDepot** option to control which apps users can see and add to Web Part pages in sites for this farm.

Set the Default Sandbox Mode for apps

The default Sandbox mode determines whether apps are automatically isolated (in a sandbox) or not (embedded) in a Web Part page. You should apply a sandbox (isolate apps) if publishing apps into the Web Part Page typically changes the layout of the Web Part Page.

To set the default Sandbox mode

1. Login is as site administrator for the site collection or specific site.
2. Click **Site Actions > Site Settings**.
3. Click **Mashup Sites for SharePoint Site Settings**.
4. Clear or set the **Default to Sandbox** option and click **OK**.

P4S Farm and Site Administration, SharePoint 2010

In addition to the basic configuration steps, SharePoint farm and site administrators can view version information for:

- P4S
- .Net Framework

This information is useful if you need to contact JackBe technical support.

For farms, open Central Administration as a farm administrator. Select **Application Management** and click **Mashup Sites for SharePoint Sites Settings** in the MashZone NextGen Administration section.

For site collections or sites, login as the site administrator and click **Site Actions > Site Settings**. Click **Mashup Sites for SharePoint Site Setting** to view this information or set other site-wide properties. See "Set the Default Sandbox Mode for apps" on page 1391 or "Disable Cookie Forwarding for MashZone NextGen Server Overrides" on page 1384 for more information.

P4S Configuration and Administration, SharePoint 2007

Once you have installed the MashZone NextGen Add-On for SharePoint (P4S) solution, configuration involves activating features across your farms and sites, setting up authentication mechanisms between SharePoint and MashZone NextGenMashZone NextGen Servers and defining connections to the MashZone NextGen Servers. See "Planning Authentication for P4S 2007" on page 1392 and "Configuring MashZone NextGen Add-On for SharePoint 2007" on page 1396 for details.

For administration tasks, see "Farm and Site Administration for P4S 2007" on page 1411.

Planning Authentication for P4S 2007

There are several options you can use to implement and optimize authentication between your SharePoint installation and MashZone NextGenMashZone NextGen Servers. This allows you to set up an appropriate level of security and user experience when your environment has bi-directional authentication between SharePoint and one or more MashZone NextGen Servers.

The challenge for user authentication with MashZone NextGen Add-On for SharePoint (P4S) 2007 is that users, who are already authenticated in SharePoint, must also authenticate with one or more MashZone NextGen Servers as seamlessly as possible. For the best user experience, a user should only be required to login to a system once and gain authorization to remote systems using their current identity.

To define the appropriate plan for authentication, you must consider the various User Experiences for P4S 2007 and Authentication Designs for P4S 2007 based on your SharePoint environment.

User Experiences for P4S 2007

There are two potential authentication experiences for users working with Web Part pages that contain mashups or apps using MashZone NextGen Add-On for SharePoint (P4S) 2007. From best experience to worst, these are:

- *Initial Challenge with SSO*: user must login once with MashZone NextGen Servers during a (browser) session in SharePoint for the first mashup or app that a user encounters in a Web Part page. No additional login challeges are issued for any other mashups or apps during that browser session. However, each new session in SharePoint requires a initial login to MashZone NextGen.
- *All Challenges:* no authentication information is shared between SharePoint and MashZone NextGen with this design. Users must login with MashZone NextGen Servers for *every* mashup or app that requires authentication. Obviously, this is the least user-friendly experience.

You choose an appropriate user experience based the "Authentication Designs for P4S 2007" on page 1393 available for your SharePoint environment.

Authentication Designs for P4S 2007

The user experiences that you can deliver using MashZone NextGen Add-On for SharePoint (P4S) 2007 depend on your SharePoint environment and how SharePoint and MashZone NextGen Servers are deployed. The factors that determine an authentication architecture include:

- *SharePoint2007 Authentication*: using Microsoft Office SharePoint Server (MOSS) 2007:
  - *With the Single Sign-On Service (MOSS SSO)* to provide an initial challenge with single sign-on experience for P4S users. Other factors in your environment also play a part in supporting single sign-on.

    The Single Sign-On Service stores user credentials for external applications, such as the MashZone NextGenMashZone NextGen Server, accessed from SharePoint. User credentials are organized by *target application*, also known as an *SSO Application Name*, allowing the Single Sign-On Service to handle credentials for different applications. You can store user credentials under one SSO Application name to provide credentials that several MashZone NextGen Servers share, allowing users to access mashups and apps from those MashZone NextGen Servers seamlessly, or keep credentials for some MashZone NextGen Servers separate using different SSO Application Names.
  - *Without the Single Sign-On Service:* this also can support an initial challenge experience depending on other factors in your SharePoint environment.

Note: P4S is not currently compatible with third-party single sign-on solutions, such as Netegrity SiteMiner, that are agent-based. If your SharePoint environment uses a third-party SSO solution, users may work in MashZone NextGen Hub with SharePoint to create mashables, mashups

and apps. However, users cannot use P4S actions in SharePoint and cannot publish mashups or apps to SharePoint.

- *SharePointand MashZone NextGenMashZone NextGen ServerDomains:* how you deploy MashZone NextGen Servers in your SharePoint environments affects the choices for authentication. If single sign-on is not possible, deploying SharePoint servers and MashZone NextGen Servers in the same domains enables the use of cookies to simplify authentication.

The combination of these factors produce three authentication architectures that you can use. The following table summarizes which architectures can be applied to specific environments and the resulting user experience:

Authentication Architecture	MOSS + SSO Service	MOSS / No SSSO Service	SharePoint / MashZone NextGen Server in Same Domains	SharePoint / MashZone NextGen Server in Different Domains	User Experience
SSO + Token Authentication	✓		✓	✓	Initial challenge
SSO + Cookie Authentication		✓	✓		Initial challenge
No SSO, Manual Authentication		✓		✓	All challenges

SSO + Token Authentication

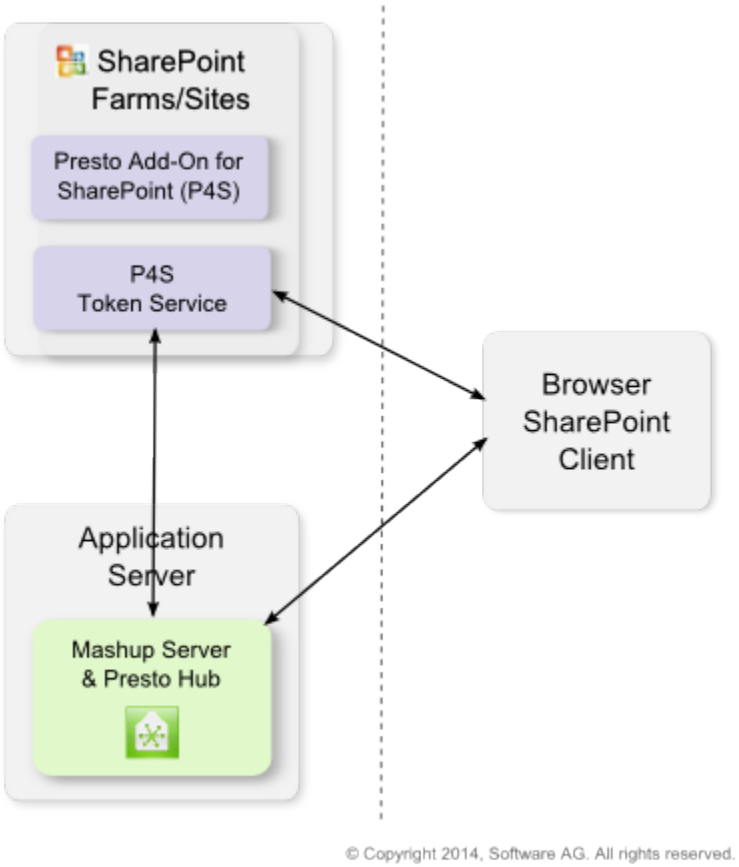
This authentication design can be used when you have chosen to use the Single Sign-On service in SharePoint to store user credentials for external applications, such as MashZone NextGen Servers. This can be used when MashZone NextGen Servers are hosted in different domains or in the same domain as SharePoint.

This solution uses the Token Service that is installed with P4S. The Token Service generates a token ID for a user's SharePoint session when they access mashups or apps. The requests sent to the MashZone NextGen Servers that host these mashups or apps include:

- A ticket with the user's token ID
- An SSO Token Server Name.

Both SharePoint and MashZone NextGen Servers must be configured with the SSO Token Server Name and the connection information to communicate with the Token

Service. MashZone NextGen Servers use this SSO Token Server Name to connect to the Token Service and redeem the user ticket and credentials.



Users receive a login challenge the first time they access mashups or apps hosted in a MashZone NextGen Server for a given SSO Application Name (a target application) in SharePoint configuration. The credentials they enter for this initial login challenge are then stored by the SharePoint SSO service under the SSO Application configured for that connection.

Note: MashZone NextGen Servers use only basic user credentials (username and password). They do not accept NTLM credentials which include Windows domains as part of the user name.

For all subsequent requests to MashZone NextGen Servers with that same SSO Application name, MashZone NextGen Servers retrieve user credentials via tokens and the Token Service and then authenticate them and retrieve user authorization information from the MashZone NextGen User Repository.

You can have each SSO Application store user credentials for one or several MashZone NextGen Server connections in SharePoint.

SSO + Cookie Authentication

This authentication design can be used when you have chosen *not* to use the SharePoint 2007 Single Sign-On service in your environment *and* the SharePoint servers and MashZone NextGen Servers for your mashup sites are located in the same domain.

With cookie forwarding, a user must login once with a MashZone NextGen Server during their current browser session. These credentials are stored as a cookie in the browser where the user is working in SharePoint. This cookie is passed by the browser to the MashZone NextGen Server in subsequent requests so no further login challenges are issued.

Note: MashZone NextGen Servers use only basic user credentials (username and password). They do not accept NTLM credentials which include Windows domains as part of the user name.

Same Domains

Domains are the same for SharePoint and MashZone NextGen Servers when the domain names fit either of these patterns:

`*.`*domain-name.type*

domain-name.type `:`*different-port*

For example, if the SharePoint server domain is `site1.com`, then a MashZone NextGen Server in the domain `ms.site1.com` or `site.com:8080` would be in the same domain.

Note: The patterns for matching domain names are a restriction of Internet browsers.

No SSO, Manual Authentication

This design is required when you have chosen *not* to use the SharePoint 2007 Single Sign-On service *and* some MashZone NextGen Servers reside in different domains than your SharePoint servers.

Because user credentials are not stored by the Single Sign-On service, MashZone NextGen Servers cannot retrieve these credentials via the Token Service. Because the SharePoint and MashZone NextGen Servers are in different domains, cookie forwarding cannot bridge this gap.

This results in manual authentication, requiring users to login to each MashZone NextGen Server as they access mashups or apps hosted by that MashZone NextGen Server to create a session or when an existing session with a MashZone NextGen Server expires.

Configuring MashZone NextGen Add-On for SharePoint 2007

You should first determine the authentication mechanisms to use with SharePoint and your MashZone NextGen Servers. See "Planning Authentication for P4S 2007"

on page 1392 for the concepts and options you need to determine an appropriate authentication plan.

Once you have an authentication plan, you complete the following steps to configure MashZone NextGen Add-On for SharePoint (P4S) 2007:

1. Activate P4S Features for SharePoint 2007
2. Configure Authentication for P4S 2007
3. Configure MashZone NextGen Server Connections in SharePoint 2007
4. Complete Miscellaneous Configuration for P4S 2007

Activate P4S Features for SharePoint 2007

MashZone NextGen Add-On for SharePoint (P4S) adds specific actions to SharePoint farms and sites. It also defines a Mashup Site template which you or users can use and adds two new Web Parts for mashups and apps respectively.

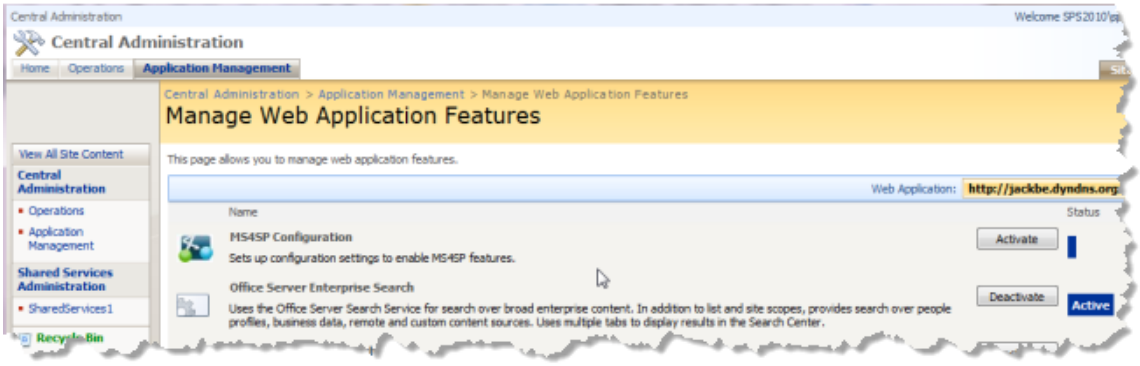
To begin configuration, you must first enable P4S features for each SharePoint farm and site collection where users should be allowed to work with mashups and apps. See these topics for instructions:

- "Enable P4S Features in Farms" on page 1397
- "Enable P4S Web Parts for Site Collections" on page 1398
- "Activate Custom P4S Actions for Site Collections" on page 1398
- "Activate the MashZone NextGen Administration Feature for Site Collections" on page 1399

Enable P4S Features in Farms

First, enable P4S features for the appropriate SharePoint farms:

1. Navigate to SharePoint Central Administration.
2. Click the **Application Management** tab and click *Manage Web Applications features*.
3. Find the MS4SP Configuration item and click *Activate* to activate these features or *Deactivate* to deactivate them.

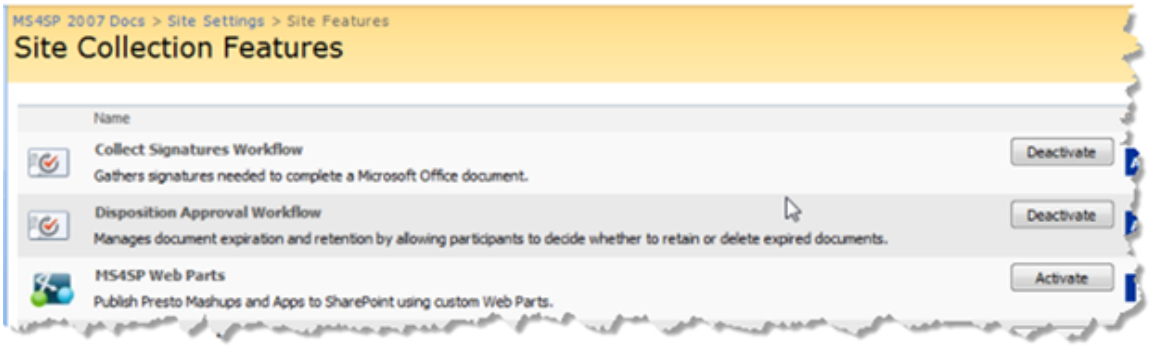


Enable P4S Web Parts for Site Collections

Enabling the P4S Web Parts feature makes both the Mashup Web Part and the App Web Part available to all sites within the site collection.

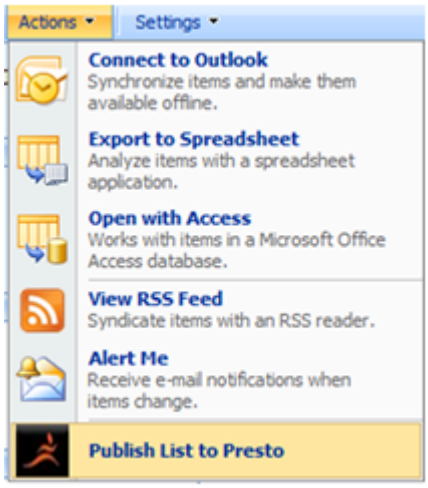
Note: Activating and deactivating these features does not affect MashZone NextGen Add-On for SharePoint features in the MashZone NextGen Server, such as registering SharePoint lists as mashable information sources or using SharePoint lists or other artifacts in mashups. You must activate the Web Parts feature, however, in order to publish apps or mashups to SharePoint from the MashZone NextGen Hub.

1. Navigate to the parent site in your site collection.
2. Click *Site Actions* > *Site Settings*.
3. Click **Site collection features**.
4. Activate the MS4SP Web Parts feature.

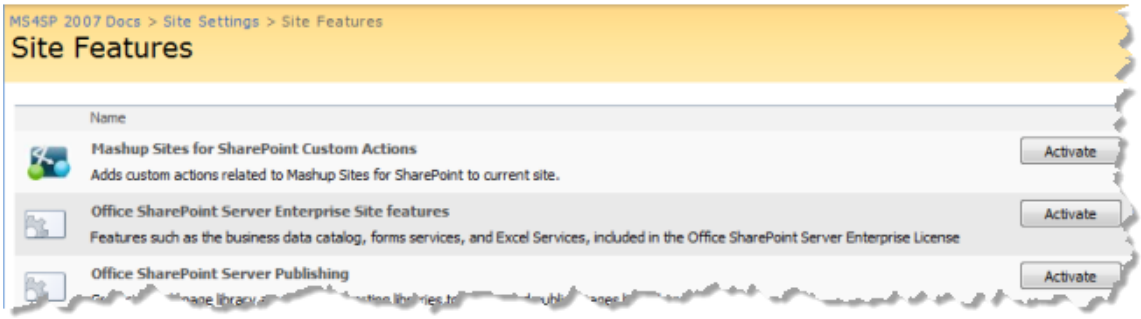


Activate Custom P4S Actions for Site Collections

In sites where it is activated, the MashZone NextGen List Action feature adds the **Publish List to MashZone NextGen** action to most SharePoint Lists in the current site. For more information, see "Register a SharePoint List as a MashZone NextGen Mashable" on page 1434.



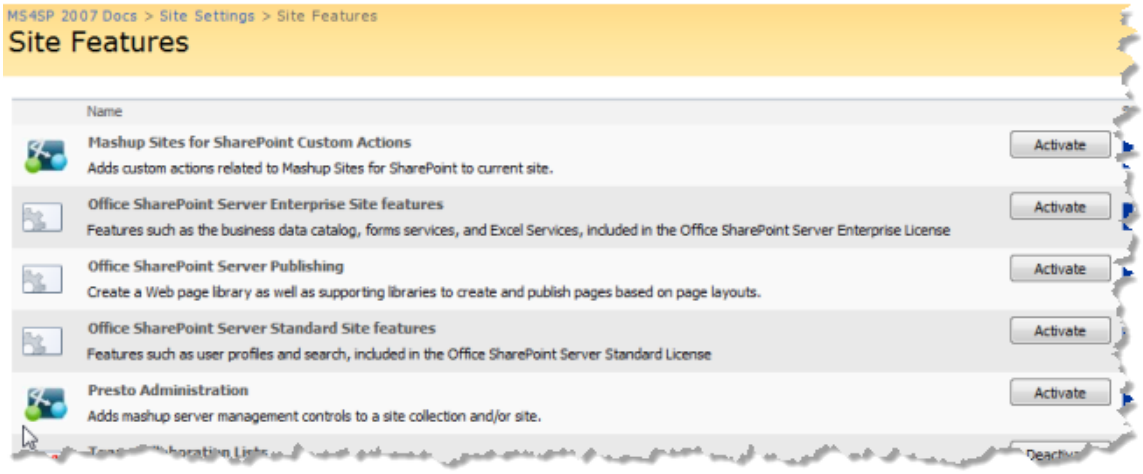
1. Navigate to the site or site collection.
2. Click **Actions > Site Settings** and click Site features.
3. Activate the Mashup Sites for SharePoint Custom Actions feature.



Activate the MashZone NextGen Administration Feature for Site Collections

Activating the MashZone NextGen Administration feature enables a site administrator to manage connections to MashZone NextGen Servers at a site level and perform other configuration and administrative tasks. For more information, see "Complete Miscellaneous Configuration for P4S 2007" on page 1410 and "Farm and Site Administration for P4S 2007" on page 1411.

1. Navigate to the site or site collection.
2. Click *Actions* > *Site Settings* and click Site features.
3. Activate the *MashZone NextGen Administration* feature.



Configure Authentication for P4S 2007

The tasks you need to complete to configure authentication between MashZone NextGen Add-On for SharePoint (P4S) and MashZone NextGen Servers depends on the authentication plan you need to implement:

- *For SSO + Token Authentication*, you use the SSO Service to manage user credentials for MashZone NextGen Servers. In addition, this uses tokens managed by the Token Service from P4S and authentication configuration in MashZone NextGen Server connections.

  Configuration steps include:

  1. If the SSO Service is not already configured, you must Configure the SSO Service in Windows and Configure the SSO Service in SharePoint.
  2. Once the SSO Service is configured, you Create SSO Applications for P4S to hold user credentials for the MashZone NextGen Servers in your environment.
  3. Then Configure SSO Token Server Names for P4S to manage user token IDs.
  4. And finally configure authentication properties for each MashZone NextGen Server connection. See "Configure MashZone NextGen Server Connections in SharePoint 2007" on page 1406 for instructions.

- *For SSO + Cookie Authentication*, you must configure cookie forwarding authentication in the properties for each MashZone NextGen Server connection. See also "Configure MashZone NextGen Server Connections in SharePoint 2007" on page 1406 for instructions.

  Cookie forwarding is enabled for user connecton overrides by default, so no additional configuration is required.

- *For No SSO, Manual Authentication*, you must:

- Clear all authentication properties in each MashZone NextGen Server connection. See also "Configure MashZone NextGen Server Connections in SharePoint 2007" on page 1406.

For more information on the requirements and user experience of these authentication designs, see "Authentication Designs for P4S 2007" on page 1393.

Configure the SSO Service in Windows

This task is required only if you are using the SSO Service in P4S authentication and the Microsoft Office SharePoint Server (MOSS) Single Sign-on (SSO) Service *is not* already running in your environment. MOSS SSO must be fully configured in Windows and SharePoint, before you can begin configuration for SSO for P4S.

Note: For a complete discussion of MOSS SSO configuration, see *Microsoft TechNet documentation* at:

- http://technet2.microsoft.com/Office/en-us/library/34d6aeca-2a18-4416-8824-85d709d1b0da1033.mspx?mfr=true
- http://technet2.microsoft.com/Office/en-us/library/cd4f4a25-e393-4e1b-9c26-a0bed175d3a21033.mspx?pf=true
- http://www.sharepointblogs.com/llowevad/archive/2007/06/25/sharepoint-2007-single-sign-on-setup.aspx

To configure the Windows service for MOSS SSO

1. From Administrative Tools, click **Services**
2. Double-click **Microsoft Single Sign-On Service**.
3. In the Log On tab of the Single Sign-on Service Properties page, click **This account**.
4. Enter the domain, user name and password for the account that you used to install SharePoint and that you use to manage your SharePoint server. **Click Apply**.

   Note: This account should be the same account used for the SharePoint application pool associated with the SharePoint sites that will be using the Single Sign-on Service. The account must have the `dbcreator` and `securityadmin` SQL Server roles on the SQL Server that will be used to host the SSO database.

5. In the **General** tab, change the startup type to `Automatic`.
6. Click **Start** and click **OK** to start this service.

Once you have configured MOSS SSO in Windows, see "Configure the SSO Service in SharePoint" on page 1402 for the remaining configuration requirements.

Configure the SSO Service in SharePoint

This task is the second part of configuration for MOSS SSO. It is required only if you are using the SSO Service in P4S authentication and the Microsoft Office SharePoint Server (MOSS) Single Sign-on (SSO) Service *is not* already running in your environment.

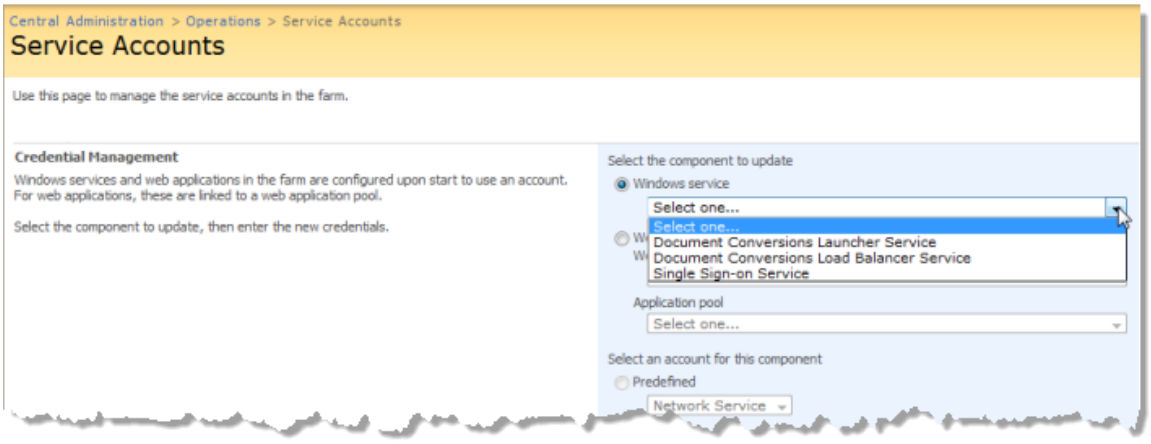
Note: For a complete discussion of MOSS SSO configuration, see *Microsoft TechNet documentation* at:

- http://technet2.microsoft.com/Office/en-us/library/34d6aeca-2a18-4416-8824-85d709d1b0da1033.mspx?mfr=true
- http://technet2.microsoft.com/Office/en-us/library/cd4f4a25-e393-4e1b-9c26-a0bed175d3a21033.mspx?pf=true
- http://www.sharepointblogs.com/llowevad/archive/2007/06/25/sharepoint-2007-single-sign-on-setup.aspx

Before you begin this task, you must "Configure the SSO Service in Windows" on page 1401.

To configure SSO in SharePoint

1. Open SharePoint Central Administration.
2. Click **Operations**.
3. Under Security Configuration, select **Service accounts**.
4. Select `Single Sign-on Service` as the **Windows service**.



5. Enter a service account username and password and click **OK**.
6. In Security Configuration, click **Manage settings for single sign-on**.
7. On the Manage Settings for Single Sign-On page, click **Manage server settings**.
8. Complete these properties:

- **Single Sign-On Administrator Account Name**: enter the same domain and user name that you used to configure the Single Sign-On service. If the user name you used to configure the Single Sign-On service is a member of a Windows security group, you can enter the name of the Windows security group instead of a user name.
- **Enterprise Application Definition Administrator Account**: enter the same domain and user name that you used to configure the Single Sign-On service.
- **Database Settings Server name**: enter the SQL Server instance name, using the *netbios\instance* naming convention, to use for the Single Sign-on database.
- **Database Settings Database name**: enter the name for the Single Sign-on database, such as SSO.
- Leave the default values for the **Ticket time out** and **Delete audit log records older than** properties.

9. Click **OK**.

At this point you should have a running instance of the MOSS SSO service including a new database for securely storing SSO user credentials. The next step is to "Create SSO Applications for P4S" on page 1403 to use in connections between SharePoint and MashZone NextGen Servers.

Create SSO Applications for P4S

If you are using SSO + Token authentication and you have the MOSS SSO service configured and running in Windows and SharePoint, you can create *SSO Applications* to use in the connections between SharePoint and MashZone NextGen Servers.

SSO Applications store user credentials for one MashZone NextGen Server or several MashZone NextGen Servers. Credentials for external MashZone NextGen Servers are associated with an Application Name that MashZone NextGen Servers use to retrieve credentials and authenticate users.

1. Open SharePoint Central Administration as a farm administrator.
2. Click **Operations**.
3. In the Security Configuration section, click **Manage settings for single sign-on**.
4. On the Manage Settings for Single Sign-On page, click **Manage settings for enterprise application definitions**.

> **Note:** If you are unable to click this link, SSO is not configured properly. See "Configure the SSO Service in Windows" on page 1401 and "Configure the SSO Service in SharePoint" on page 1402 for instructions.

5. Click **New Item** and set the following properties:
   - **Display Name**: the name of the SSO application to show users.
   - **Application Name**: the SSO application ID.

User credentials stored with this SSO application ID are used by all MashZone NextGen Server connections that are configured with this SSO Application.

- **Contact e-mail address**: an email address for the administrator who should manage SSO applications.
- **Account type**: the type of account that supplies credentials. Typically this is set to `Individual` indicating that each user has their own credentials.
- **Authentication type**: make sure the `Windows authentication` option is clear.
- For Logon Account Information, clear all fields except **Username** and **Password**.
- Click **OK**.

Configure SSO Token Server Names for P4S

Farm administrators can view, create, update, or remove *SSO Token Server Names* that are used when SSO + Token Authentication is used for connections to MashZone NextGen Servers.

SSO Token Server Names are placeholders that are substituted for the host portion, such as `http://sharepoint.myorg.com`, of the URL for a SharePoint MOSS SSO server. P4S sends the SSO Token Server Name when users connect to a MashZone NextGen Server. The MashZone NextGen Server uses this name to validate user credentials with the named MOSS SSO server. See "SSO + Token Authentication" on page 1394 for more information.

To manage SSO Token Server Names

1. Navigate to Central Administration and click **Operations**.
2. On the Operations page, go to the MashZone NextGen section and click **Manage SSO Token Server Names**.

   Note: MashZone NextGen Servers must also have configuration for connections to SharePoint site collections with a matching SSO Token Server Name to resolve the placeholder value sent from SharePoint. For more information, see "Configure MashZone NextGen Connections to SharePoint" on page 1655.

3. Click **New SSO Token Server Name** to add configuration for a new token server. Complete these properties:
   - **SSO Token Server Name**: enter a name to identify this token server. This will be the replacement value added to URLs to MashZone NextGen Servers.
   - **SSO Token Server URL**: enter the domain and port number for this token server.
4. Then click **OK**.

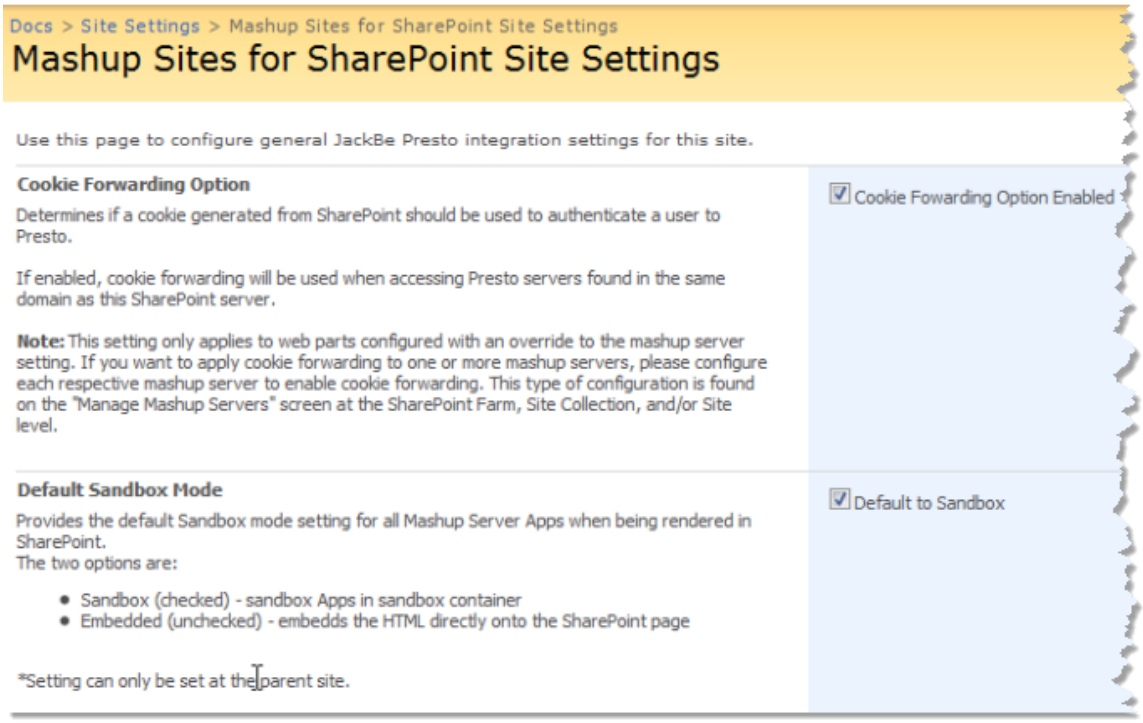
You can also edit or delete existing token servers from this page. Click the pull-down menu next to the token server you want to edit or delete and select the corresponding action from the menu.

Disable Cookie Forwarding for MashZone NextGen Server Overrides for Sites

When users choose the preconfigured MashZone NextGen Server connection in Web Parts, all aspects of the connection including authentication is defined in the connection. If users override this preconfigured connection, cookie forwarding authentication is enabled by default.

If you are using No SSO + Manual authentication, you should disable cookie forwarding for user overrides of preconfigured MashZone NextGen Server connections. You do this in site settings for a site collection or a specific site.

1. Login is as site administrator for the site collection or specific site.
2. Click **Site Actions > Site Settings**.
3. Click **Mashup Sites for SharePoint Site Settings**.
4. Clear the **Cookie Forwarding Option Enabled** option and click **OK**.



Configure MashZone NextGen Server Connections in SharePoint 2007

MashZone NextGen Add-On for SharePoint (P4S) 2007 allows you to connect multiple MashZone NextGen Servers to many SharePoint sites and site collections, based on your P4S license. After a connection is configured in both MashZone NextGen and SharePoint, users can publish mashups and apps to SharePoint as native Web Parts. Users can also use information from SharePoint as mashable information sources in MashZone NextGen for mashups or apps.

For each connection, the visibility of the mashups and apps hosted in a MashZone NextGen Server within SharePoint and the visibility of SharePoint information within a MashZone NextGen Server is based on:

- Where you define the connections in SharePoint.

  Farm administrators or site administrators can configure MashZone NextGen Server connections to farms, site collections or individual sites. You can also define "Duplicate Connections to MashZone NextGen Servers" on page 1409 if needed.

  Users who publish apps or mashups to Web Parts in SharePoint can also override the default connections for a specific Web Part. This ensures that users can control settings at a granular level if needed. For more information about overriding MashZone NextGen Server connections in Web Parts, see "Update Mashup Web Part Properties" on page 1424 or "Update App Web Part Properties" on page 1441.

- How connections to SharePoint are defined in the MashZone NextGen Server.

For a discussion of the common patterns for configuring connections between SharePoint and MashZone NextGen Servers, see "Configure MashZone NextGen Connections to SharePoint" on page 1655.

Then Add a MashZone NextGen Server Connection to SharePoint 2007 or Edit a MashZone NextGen Server Connection as needed.

Add a MashZone NextGen Server Connection to SharePoint 2007

The scope of the connection that you create is determined by your account and the farm or site where you add the connection.

Visibility for connections can also be affected by where you add the connection. MashZone NextGen Server connections defined for site collections are immediately visible while visibility at sites and site collections for MashZone NextGen Server connections defined for farms may *be delayed* up to 30 minutes because of caching.

1. Verify or complete the following *required* prerequisites:

   a. Install the MashZone NextGen Server, if you have not already done so. See "Install or Upgrade MashZone NextGen Servers for P4S" on page 1372 for instructions.

   b. Make sure the MashZone NextGen Server is running. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

c. Configure connections in the MashZone NextGen Server to the appropriate SharePoint farms, site collections and sites. See "Configure MashZone NextGen Connections to SharePoint" on page 1655 for instructions.

2. To create a connection to a MashZone NextGen Server in a farm:

   a. Open SharePoint Central Administration as a farm administrator.

   b. Click **Operations** and click **Manage Mashup Servers** in the MashZone NextGen Administration category. screen

   c. Skip to Step 4.

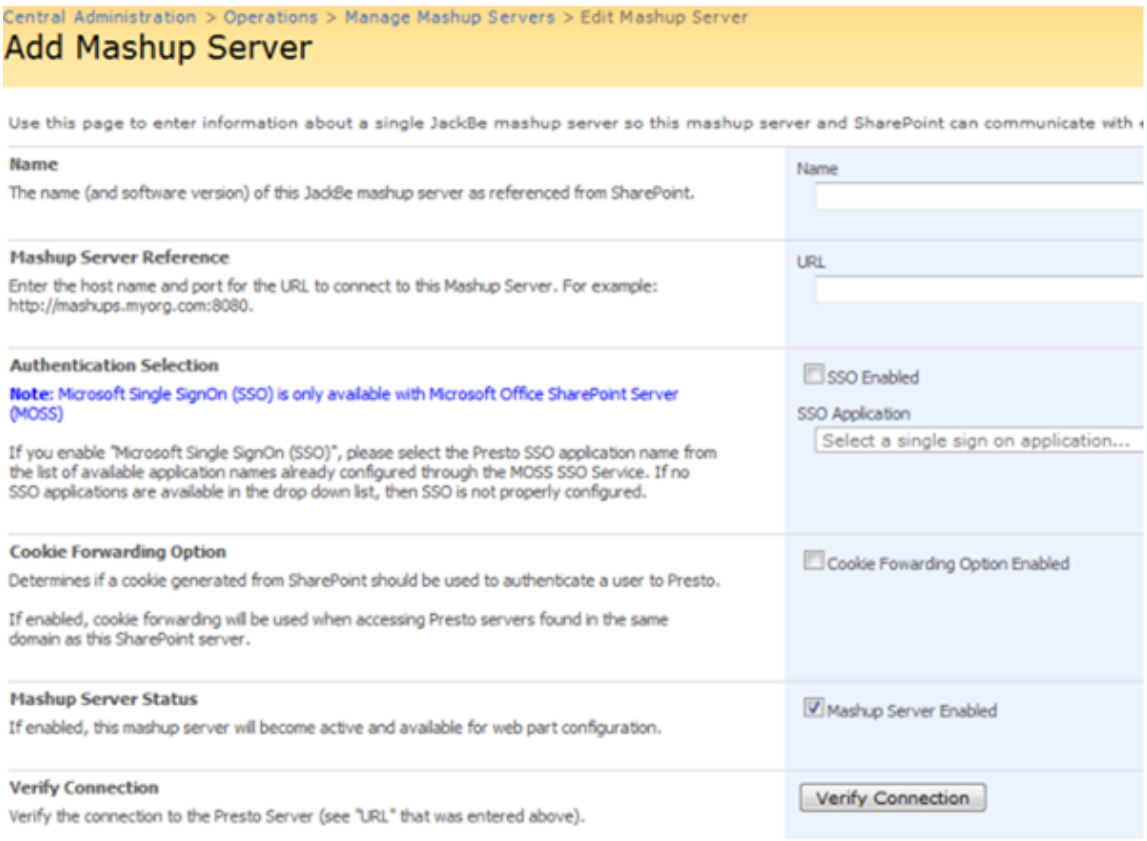
3. To create a connection to a MashZone NextGen Server in a site collection or a specific site:

   a. Login is as site administrator for the site collection or specific site. If needed navigate to the home page for the site you want.

   b. Click **Site Actions > Site Settings**.

   c. Click **Manage MashupServers** in the MashZone NextGen Administration category.

4. Click **New Mashup Server** from the Manage Mashup Server page. The Add Mashup Server page opens.



5. Complete basic properties for this connection:

   - **Name:** choose a name that describes the location or purpose of the MashZone NextGen Server you are connecting to.
   - **URL:** enter the domain name and port, or IP address and port, for the URL to connect to this MashZone NextGen Server. For example: `http://mashups.myorg.com:8080` or `http://204.10.1.101:8080`.

   Note: You may omit the port number in this URL if the MashZone NextGen Server uses port 80.

   URLs cannot use the `localhost` domain name. Use the IP address instead.

6. Set authentication properties for this connection based on your authentication plan. See "Authentication Designs for P4S 2007" on page 1393 for more information on the types of authentication you can use.

   If you are using SSO + Token authentication, set these properties:

   - **SSO Enabled:** set this option to enable single sign-on authentication. You must use the MOSS SSO Service with your SharePoint 2007 Server to provide single sign-on for MashZone NextGen Server connections.
   - **SSO Application:** if you have enabled SSO (see above), choose the appropriate target application to use for authentication of credentials when connecting to this MashZone NextGen Server.

     If no SSO applications are listed, then SSO is not properly configured. See "Configure Authentication for P4S 2007" on page 1400 for more information to correct this problem.

   If you are using SSO + Cookie authentication, set this property:

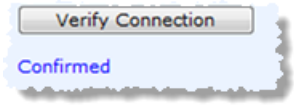
   - **Cookie Forwarding Enabled:** set this option to allow cookies to handle authentication credentials between SharePoint servers in this farm and this MashZone NextGen Server. The MashZone NextGen Server*must* be in the same domain as this SharePoint farm.

   If you are using No SSO, Manual authentication, clear all the SSO and Cookie properties.

7. Set the **Mashup Server Enabled** option to activate this connection and click **Verify Connection.**

   If the connection fails, check to make sure the URL you entered is correct and the MashZone NextGen Server you are connecting to is running. See "Start and Stop the MashZone NextGen Server" on page 1570 for startup instructions.

   A Confirmed message displays once your connection is verified.

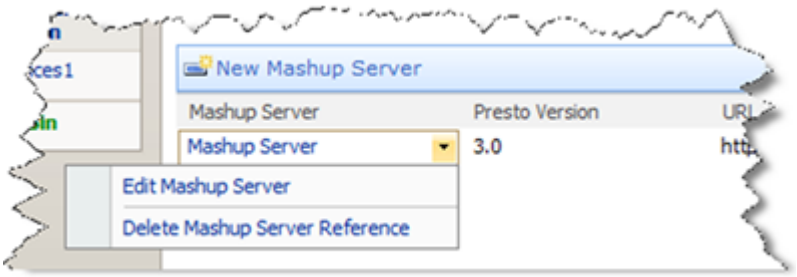


8. Once you have verified the connection, click **OK**. The Manage Mashup Servers page appears, listing your new MashZone NextGen Server connection.

Edit a MashZone NextGen Server Connection

1. Find the MashZone NextGen Server Connection based on the scope of the connection:
   - For connections with a farm scope, open SharePoint Central Administration as a farm administrator. Click **Operations** and click **Manage Mashup Servers** in the MashZone NextGen Administration category.
   - For connections with a site collection or site scope, login is as site administrator for the site collection or specific site. Click **Site Actions > Site Settings**. Click **Manage Mashup Servers** in the MashZone NextGen Administration category.

   The Manage Mashup Servers page opens listing all existing connections in this scope.
2. Click the drop-down menu for the MashZone NextGen Server connection you need to edit and select **Edit Mashup Server**.



3. Update the properties you need to edit and click **Verify Connection**.
4. Once the connection is confirmed, click **OK**.

Duplicate Connections to MashZone NextGen Servers

You can define duplicate MashZone NextGen Server connections with different scopes, such as defining one connection for a SharePoint farm and a duplicate connection for a specific site collection in that farm. This can be useful, for example, if you need to configure a different authentication mechanism for a specific site collection.

When duplicate connections are defined, the site-collection-specific connection overrides the farm-level connection for that specific site collection. All other site collections inherit the farm-level connection.

Complete Miscellaneous Configuration for P4S 2007

There are two additional configuration options that you can set for farms, site collections or individiual sites that pertain to apps:

- Set the Scope of Apps in the App Gallery allows you to have finer control over which apps users can choose to add to App Web Parts from the App Gallery.
- Set the Default Sandbox Mode for Apps allows you to determine whether apps are automatically isolated (in a sandbox) or not (embedded) in App Web Parts. This can affect performance, but also may affect style and JavaScript interactions with the Web Part page.

Set the Scope of Apps in the App Gallery

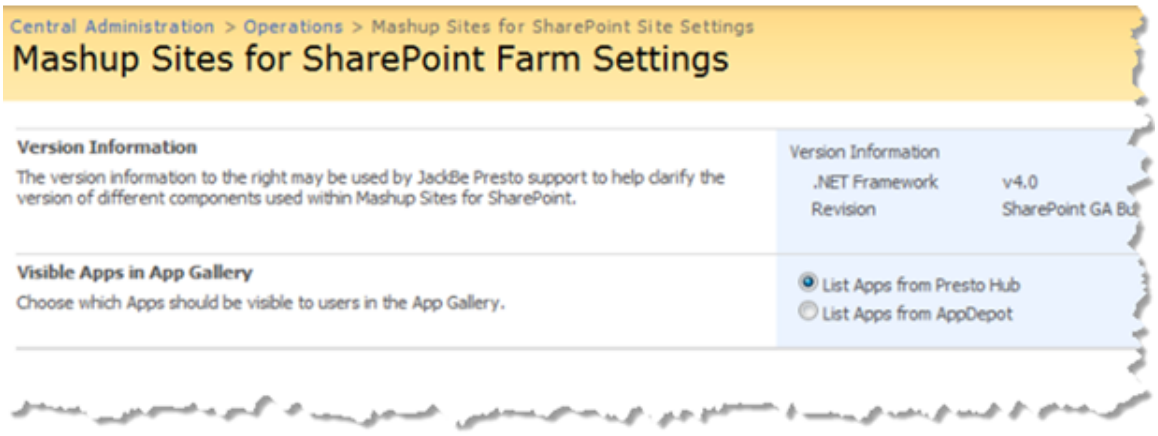
You have an option for SharePoint farms to determine whether 'production' apps or 'development' apps are listed in the App Gallery when users publish apps to Web Part pages.

When users create apps in the MashZone NextGen Hub, they may not be completely finished or 'production-ready' for all users to work with. Once apps are complete, users publish apps to the MashZone NextGenAppDepot where they are reviewed and approved by an AppDepot administrator. Published apps are now 'production-ready' and available for other users to work with.

By default, the App Gallery shows SharePoint users all the apps from the MashZone NextGen Hub. This allows users to build or update Web Part pages based on newly developed apps. To ensure that users only see 'production-ready' apps in a SharePoint farm, you can instead limit the App Gallery to those apps which have been pulished in the AppDepot.

To control app visibility in the App Gallery for all sites in a SharePoint farm

1. Open Central Administration as a farm administrator and click **Operations**.
2. On the Operations page, go to the JackBeMashZone NextGensection and click **Mashup Sites for SharePoint Site Settings**. This opens the MS4SP Farm Settings page.



3. Set either the **List Apps from** MashZone NextGen Hub or **List Apps from AppDepot** option to control which apps users can see and add to Web Part pages in sites for this farm.

Set the Default Sandbox Mode for Apps

The default Sandbox mode determines whether apps are automatically isolated (in a sandbox) or not (embedded) in a Web Part page. You should apply a sandbox (isolate apps) if publishing apps into Web Part Pages tends to change the layout of the Web Part Page or causes JavaScript issues.

To set the default Sandbox mode

1. Login is as site administrator for the site collection or specific site.
2. Click **Site Actions > Site Settings**.
3. Click **Mashup Sites for SharePoint Site Settings**.
4. Clear or set the **Default to Sandbox** option and click **OK**.

Farm and Site Administration for P4S 2007

In addition to the basic configuration steps, SharePoint farm and site administrators can find version information for:

- The MashZone NextGen Add-On for SharePoint (P4S) solution
- .Net Framework

This information is useful if you need to contact Software AG technical support.

Version Information for Farms

To view version information for farms

1. Open Central Administration as a farm administrator and click **Operations**.
2. Click **Mashup Sites for SharePoint Site Settings** in the MashZone NextGen section.

Version Information for Sites

To view version information for site collections or sites

1. Login as the site collection or site administrator.
2. Click **Site Actions > Site Settings**.
3. Click **Mashup Sites for SharePoint Site Settings** in the MashZone NextGen Mashup Sites for SharePoint section.

SharePoint Credentials for Mashables and Mashups

The mashable information sources or mashups in MashZone NextGen that work with information from SharePoint use SharePoint credentials to retrieve information when

they are run. The SharePoint credentials that are used when a mashable or mashup is run can be either:

NTLM or Basic Authentication	SSO Authentication
The current user's SharePoint credentials. This is used in mashables when both: ■ The user has created MashZone NextGen User Attributes with their SharePoint credential information in their MashZone NextGen Hub profile. See "Add or Manage SharePoint Credentials in Your MashZone NextGen Hub Profile" on page 1413 for instructions. ■ The mashable was configured to use the current's user credentials when it was registered. If the mashable has been configure to use the current user's credentials (the MashZone NextGen User Attributes), but the user has not created these credentials in MashZone NextGen, either: ■ The service account credentials for this connection are used instead, when the connection is configured to use default credentials. ■ The connection fails if the connection is configured to use *only* user credentials.	An HTTP header with user identity information that is validated by an agent-based single sign-on (SSO) solution. With SSO authentication, you must Add a Global Attribute for Agent-Based SSO Authentication with SharePoint to identify the HTTP headers or request parameters that provide user identity information. The MashZone NextGen Server uses this attribute to determine which HTTP headers or parameters to forward to SharePoint when mashables or mashups are run.
The credentials supplied when the mashable was registered or the mashup was created. These credentials can be either: ■ Credentials for the user that registered the mashable or created the mashup. ■ The service account credentials configured for the connection to this SharePoint server if the connection is	

NTLM or Basic Authentication	SSO Authentication
configured to use default credentials or to use *only* the service account. Each SharePoint server connection creates MashZone NextGen global attributes under that server name with these service account credentials.	

Add or Manage SharePoint Credentials in Your MashZone NextGen Hub Profile

Add SharePoint Credentials

Users can add or update their SharePoint credentials in MashZone NextGen when they register mashables that connect to SharePoint or when they create mashups using SharePoint blocks in Wires. These credentials are saved as MashZone NextGen user attributes in each user's profile in the MashZone NextGen Hub.

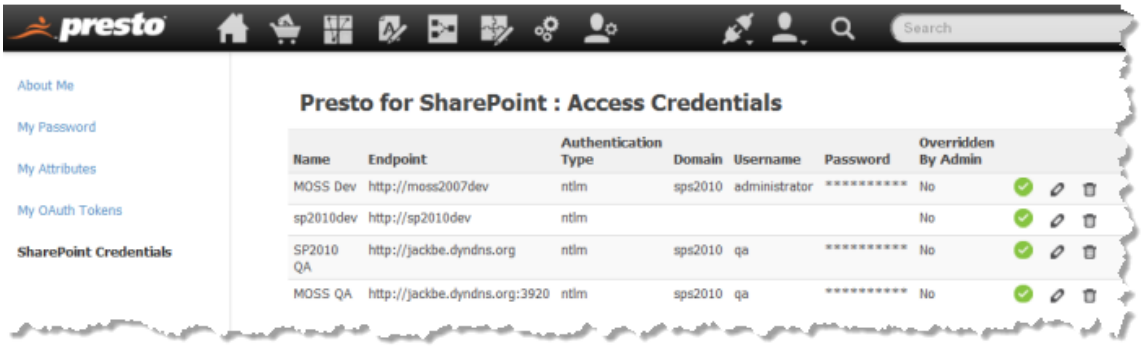
You use the **Credentials** section in SharePoint blocks or Register SharePoint List to add or manage these credentials. See "Register SharePoint Lists" on page 156 or the SharePoint blocks in "Add Actions or Other Blocks" on page 297 for instructions.

You can also manage your SharePoint credentials for all SharePoint servers in your MashZone NextGen Hub profile.

Manage SharePoint Credentials

SharePoint credentials are added to your MashZone NextGen Hub profile when you register mashables that connect to SharePoint or create mashups using SharePoint blocks in Wires. You can also view and manage your SharePoint credentials for all SharePoint servers in their MashZone NextGen Hub profile:

1. Click *Your Account* > *Profile* in the MashZone NextGen Hub menu bar.
2. Click **SharePoint Credentials** and select credentials for a SharePoint server.



This lists the credentials that you have defined for specific SharePoint servers along with specific information for that connection. MashZone NextGen administrators

can also configure SharePoint connections to use only the service account for the connection, thus overriding any credentials you may have for have added for that SharePoint connection.

3. To update these credentials, click **Edit** and update the **Username**, **Password** or **Domain** properties. Click **Saves changes**.
4. To delete these credentials, click **Delete**.

Add a Global Attribute for Agent-Based SSO Authentication with SharePoint

When authentication for MashZone NextGen Server connections to SharePoint use an agent-based SSO solution, such as SiteMinder, authentication information is supplied by the SSO solution in HTTP headers in the request sent to the MashZone NextGen Server. These headers must be included in MashZone NextGen Server requests to SharePoint. You define the HTTP headers that may be used for SSO authentication by *any*SharePoint connection in a MashZone NextGen global attribute.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Attributes section and click **Global Attributes**.
3. Click **Add new global attribute**.
4. Enter `sp-ssoheaders` as the attribute name.
5. In the **Attribute value** field, enter the names of all HTTP headers or request parameters that may be used by the agent-based SSO solution to pass authentication information. Separate HTTP header or parameter names with commas, but *no spaces*. For example:

   ```
   SM_USER,SM_ALT_USER
   ```

   This list includes SSO HTTP headers or parameters for all connections to SharePoint.
6. Click **Save**.

See "Configure MashZone NextGen Connections to SharePoint" on page 1655 for more information about SharePoint connection configuration.

MashZone NextGen Add-On for SharePoint User Guide

Mashups and apps enable your sites in SharePoint to display dynamic, up-to-the-minute, integrated data that is specifically relevant to your organization, suppliers, customers or other partners. SharePoint Web Part Pages allow you to construct dashboards from many different types of content to give viewers a coherent and multi-faceted source of information.

Users easily create mashups and apps using simple, graphical tools in MashZone NextGen that build on many different *information sources* such as web services, web feeds, databases or existing systems in your organization.The MashZone NextGen Add-On for SharePoint allows you to easily incorporate these mashups and apps as native Web Parts in your SharePoint sites, providing deep integration with other SharePoint content using SharePoint's Web Part communication framework. For mashups, you

have additional integration options to publish them directly as native SharePoint Lists or integrate them dynamically as external Lists using the Business Connectivity Service (BCS).

- *Mashups* provide a powerful way to gather relevant data for research and analysis. You have three choices on how to publish mashups that affect the features available in SharePoint:

Mashup Published	Shows Dynamic Data	Users Can Change Input Parameters or Properties	Accessible to SharePoint Services
In Web Parts Mashups published in Web Parts allow you to use mashup data on a Web Part Page in an easily readable table format, similar to a list view. See "Publishing Apps and Mashups to Web Parts" on page 1416.	✓	✓	
As native SharePoint Lists See "Publish Mashup Results as Native SharePoint Lists" on page 1427.			✓
As SharePoint External Lists See "Publish Mashup Results as External SharePoint Lists" on page 1428.	✓		✓

With Web Parts or External Lists, changes in mashup results are visible in real-time in SharePoint. With native Lists, the data is just a snapshot of mashup results when the mashup was published.

With Web Parts, users that have edit permissions have added flexibility to change input parameters for the mashup and see different data. They can also edit Web Part properties for the mashup.

With native or external Lists, data within the mashup results is handled just like any SharePoint List. Users can search through the data or use this with other SharePoint services, such as Performance Point.

- *Apps* combine mashups or other information sources with one or more *views* of that information including charts, tables and maps. Different views can convey information in more accessible, effective, and memorable formats.

  You can also use apps to connect to a remote web service to view, update, add, or remove data. For instance, you could input travel expenses in a simplified form.

  Apps can also be combined to work together in dashboards or workspaces. Apps are also easily shared with others in the MashZone NextGen Enterprise AppDepot, Enterprise Java Portals, SharePoint or any web page.

  See "Publishing Apps and Mashups to Web Parts" on page 1416.

Publishing Apps and Mashups to Web Parts

The basic tasks you complete to add apps or mashups to SharePoint pages using Web Parts are:

- Create a Web Part Page.
- Add a Mashup Web Part to a Web Part Page or Add an App Web Part to the Web Part Page .
- Publish a Mashup to a Mashup Web Part or Publish an App to an App Web Part

Other Tasks

For mashups, you can Publish Mashup Results as Native SharePoint Lists or Publish Mashup Results as External SharePoint Lists instead.

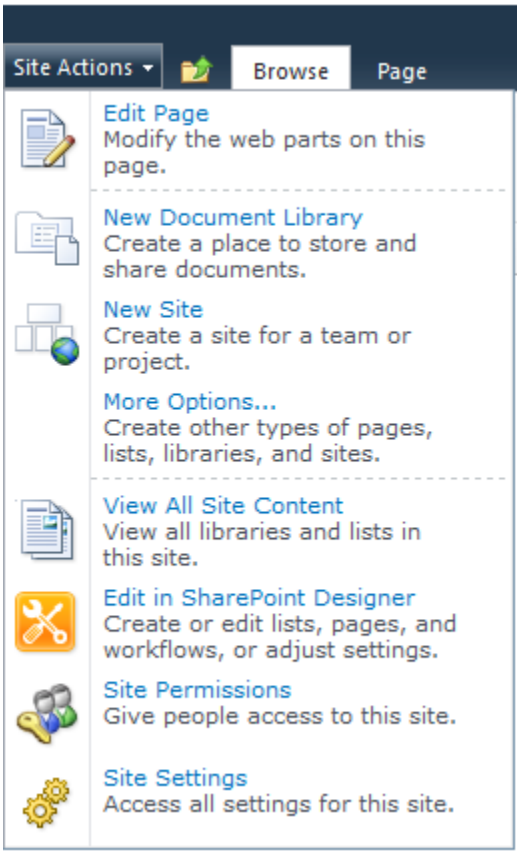
In addition to adding mashups or apps to Web Part Pages in SharePoint, you can also use information in SharePoint lists in MashZone NextGen as a mashable information. To use list information in MashZone NextGen, see "Register a SharePoint List as a MashZone NextGen Mashable" on page 1434.

Create a Web Part Page

You control and customize the layout and content of a Web Part page. A page is composed of one or more Web Parts that may contain apps, mashups or other information from inside and outside of SharePoint.

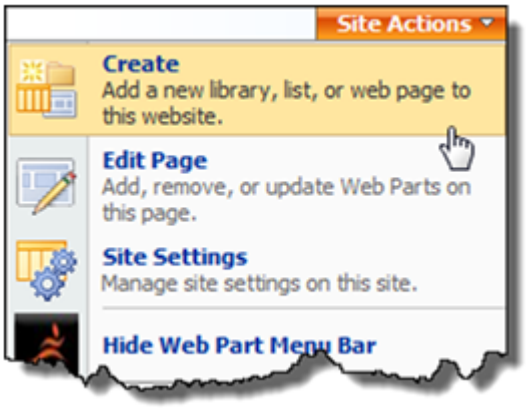
To create a Web Part page

1. In SP 2010, click **Site Actions > More Options.** The Browse Actions window opens.



Then filter actions by **Page**, choose **Web Part Page** and click **Create**.

2. In SP 2007, click **Site Actions > Create**.



Then select **Web Part Page** under Web Pages.

3. Name your Web Part Page and set the Overwrite option if this page should overwrite an existing page with the same name in the document library. This is the file name that appears in headings and links throughout the site.

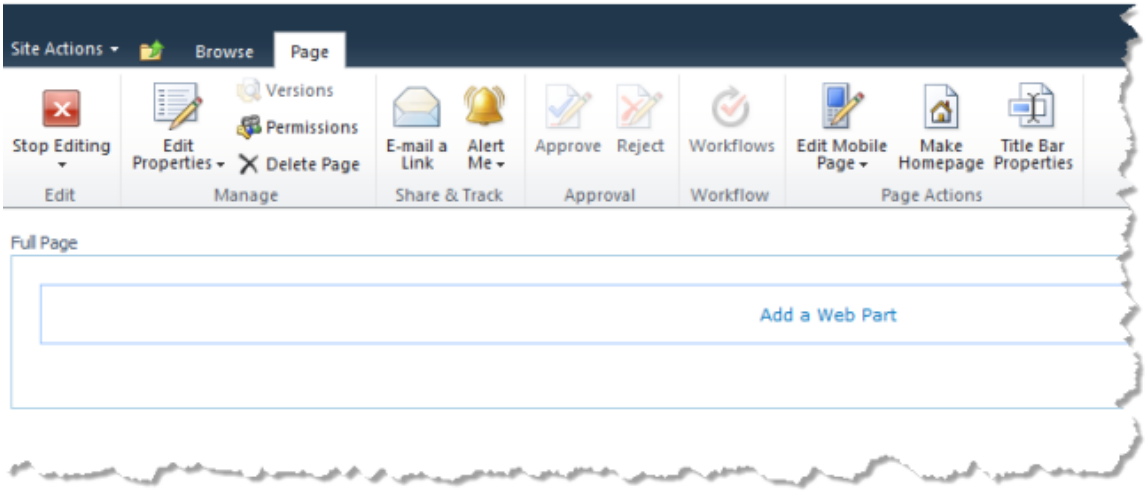
4. Select a layout from the **Choose a Layout Template** list. A preview of the selected layout appears to the left.

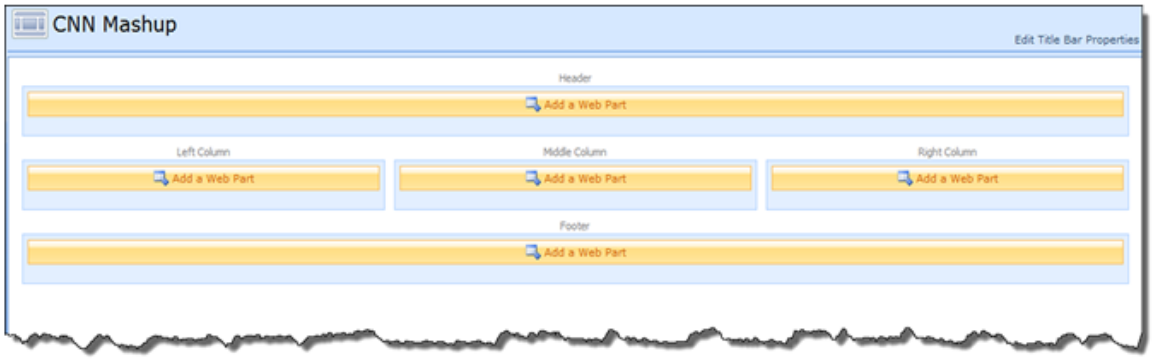
5. Choose the **Document Library** where you want the Web Part Page to be saved.

> **Important:** Do not save Web Part pages in a Document Library that is configured to require checkout before modifications. You cannot add mashups or apps to Web Part Pages in this case.

6. Click Create.

   The new Web Part page opens in Edit Mode with zones displayed, based on the layout you selected, where you can add Web Parts.





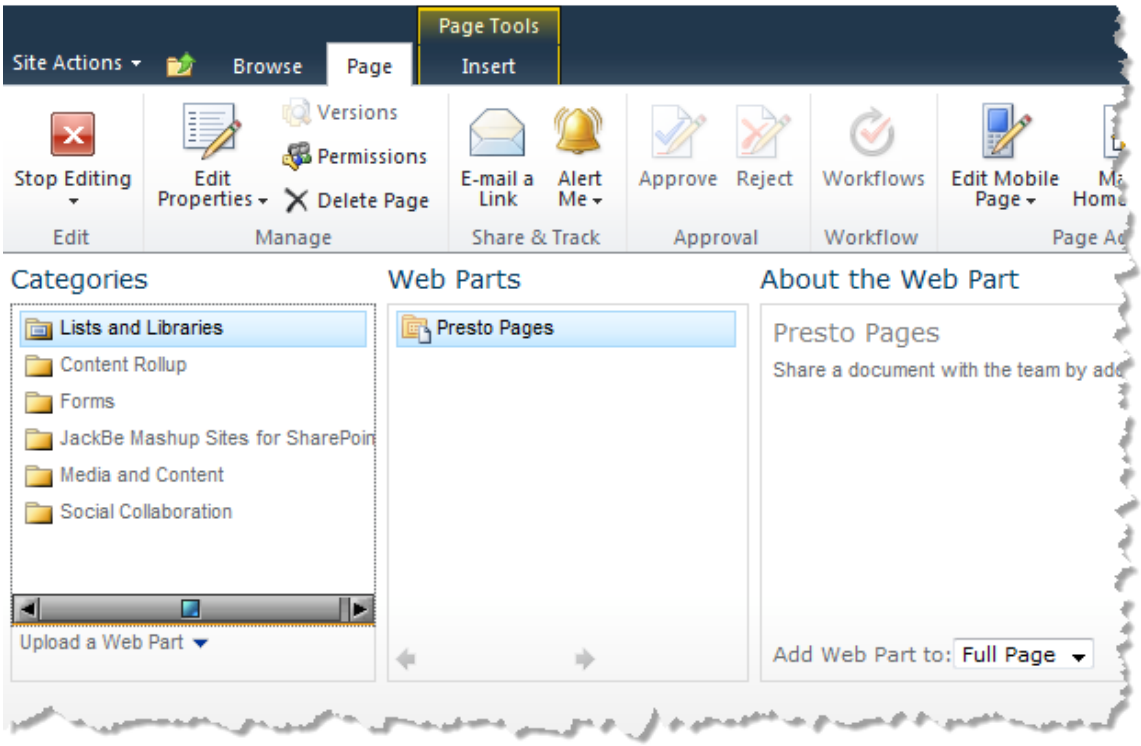
In Edit Mode, you can use the toolbar that appears above the page layout to edit page properties, delete the page, edit title properties or perform other actions. Use **Edit Title Bar Properties** to change the page title or add a caption, description, or image.

You can now Add a Mashup Web Part to a Web Part Page or Add an App Web Part to the Web Part Page and then publish mashups or apps to these Web Parts.

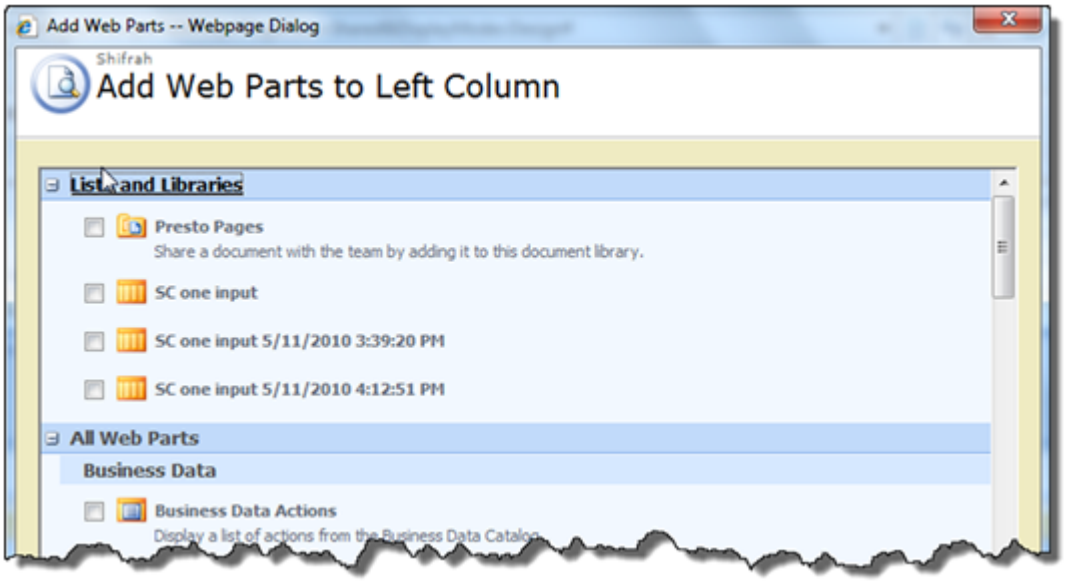
Add a Mashup Web Part to a Web Part Page

In order to publish a mashup to a Web Part Page, you must add a Mashup Web Parts to a zone in that page. You then publish mashups to each Mashup Web Part to display data from those mashups in table format.

1. If the page where you want to place a mashup is not already in edit mode, edit the page.
2. Click **Add a Web Part** in the zone on your Web Part page where you would like to place a mashup. In SP 2010, Web Part filters open:



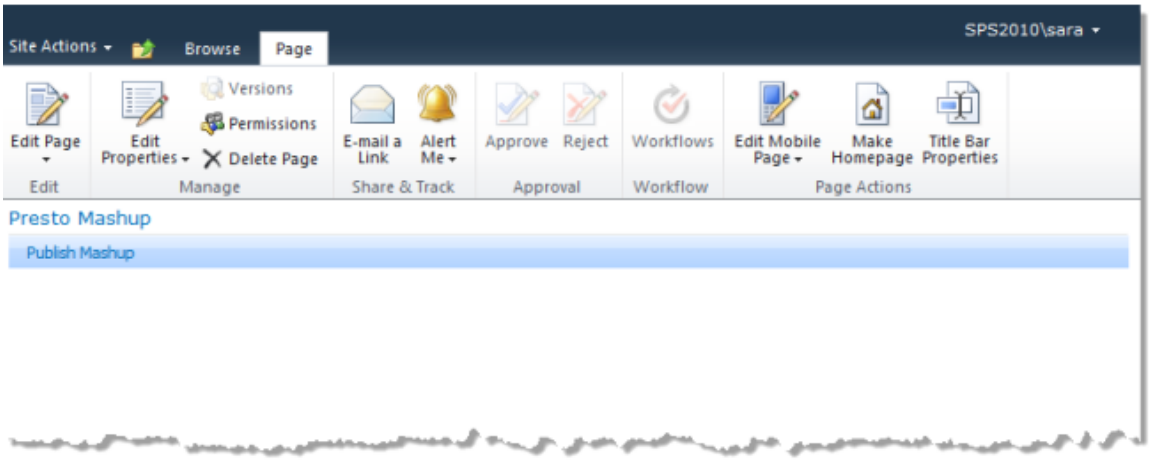
In SP 2007, the Add Web Parts menu opens:



3. Select the **JackBe Mashup Sites for SharePoint** category and click **Mashup Web Part**.
4. Click **Add** (in the bottom right corner).
5. Continue editing this Web Parts page, as needed. You can:
   - Add appropriate Web Parts to other zones on the page.

- Edit the properties for your Mashup Web Part or for any Web Part on the page. See "Update Mashup Web Part Properties" on page 1424 for more information.
- Select and drag Web Parts to move them to other zones.

6. Once you are finished adding and modifying Web Parts, click **Stop Editing** in the Web Part page ribbon. The page returns to the normal display mode, with a Mashup Web Part and toolbar in the zones you have selected.



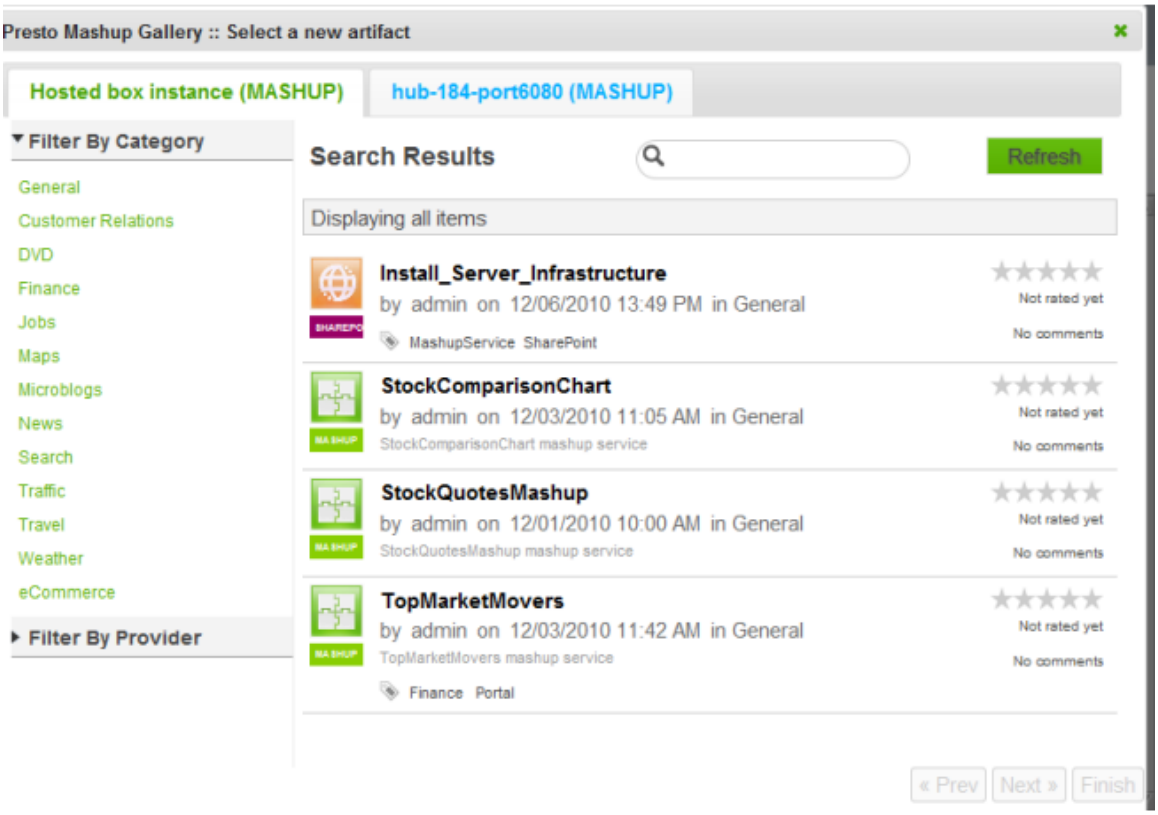
Publish a Mashup to a Mashup Web Part

Once you have stopped editing your Web Part page, you can select the specific mashup to use in any Mashup Web Part on your page from the Mashup Gallery.

1. Click **Publish Mashup** from the toolbar in any of the Mashup Web Parts on your page.
2. If prompted, enter credentials to log in to the MashZone NextGen Servers you want to work with. You can also **Skip** specific MashZone NextGen Servers. Mashups from the MashZone NextGen Servers that you skip will not be listed.

   Note: Do *not* include the Windows domain in your username, such as `mydomain\myname`, when you log in. The MashZone NextGen Server does not accept domain names. You may also need to add MashZone NextGen Servers to your list of trusted sites or change your cookie preferences to allow session cookies. If you still receive login errors, check with your MashZone NextGen administrator.

   The Mashup Gallery opens displaying a list of mashups. This list is cached and refreshes periodically. Use **Refresh** to manually refresh this list.

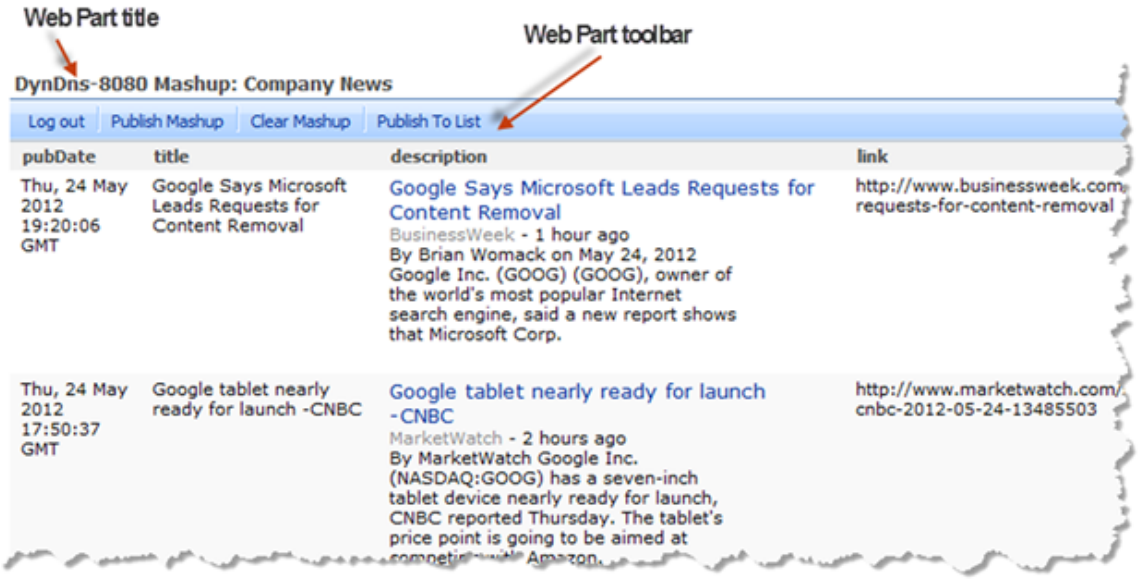


3. Find and select the mashup you want to add to this Web Part.

   You can filter the list of available mashups by category, by provider or enter a partial name in the search field.

4. If the mashup has input parameters, update these parameters as needed.

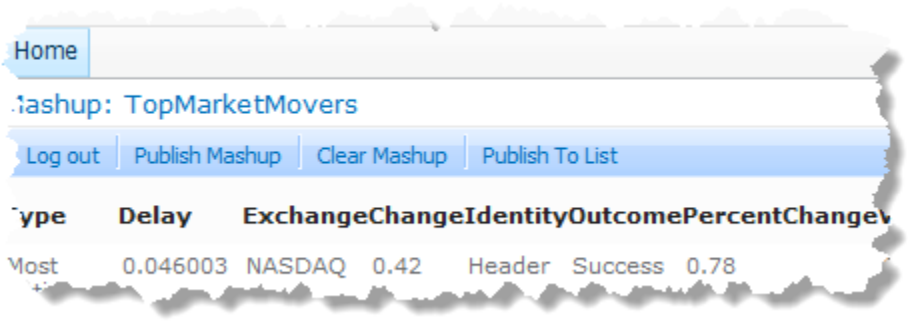
5. Click **Finish**.



The mashup you selected displays in the zone for this Web Part and, if you have appropriate permissions, includes a toolbar with actions for this mashup. See "Mashup Web Part Toolbar" on page 1423 for more information. See also "Update Mashup Web Part Properties" on page 1424 for other tasks you can perform, such as changing input parameters.

Mashup Web Part Toolbar

If you have permission to edit a Web Part Page, the mashup displays a toolbar with editing actions. The actions that appear in the toolbar change based on the status of the Web Part.



Actions on this toolbar include:

- **Log Out**: ends your session with the MashZone NextGen Server for this mashup. You can then log back in using different credentials. Simply click **Log Out**, enter another username and password and click **Log In**.
- **Publish Mashup**: allows you to find and add a different mashup to this Mashup Web Part. See "Publish a Mashup to a Mashup Web Part" on page 1421 for instructions.

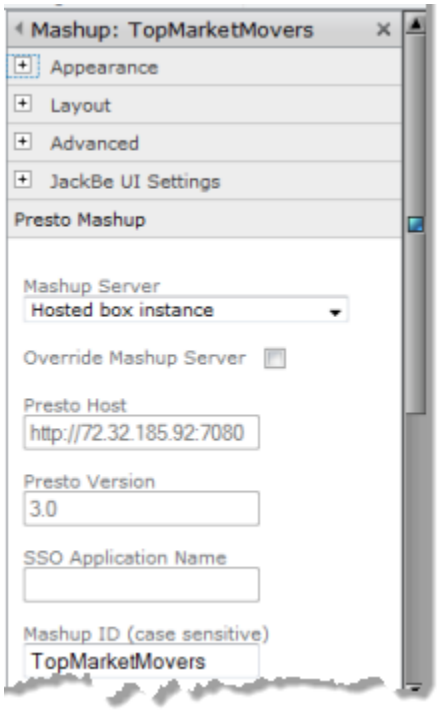
- **Clear Mashup** removes the current mashup in this Mashup Web Part. Use **Publish Mashup** to find and add a different mashup.
- **Publish to List**: creates a SharePoint list in the current site with a snapshot of the data currently shown in this mashup. This creates a view named `MashZone NextGen` for this new SharePoint list. The title for the new list is the mashup's name. If a list with that title already exists in the current site, a Date/Time stamp is added to distinguish the two.

Update Mashup Web Part Properties

You can update Web Part properties to make changes to the appearance, layout and advanced features of the Web Part. Changes you make to the Web Part properties are visible to anyone viewing the published Web Part Page.

To edit Mashup Web Part properties, click the down arrow in the right corner of the Mashup Web Part title bar and select **Edit Web Part**.

This puts the page in Edit Mode and opens the Web Part Properties pane for this mashup:



In addition to the standard Web Part properties, Mashup Web Parts have two additional categories or properties: JackBe UI Settings andMashZone NextGenMashup. Use the properties in these categories to:

- "Change the Web Part Title" on page 1425
- "Change the Mashup Server Connection for a Mashup Web Part" on page 1426
- "Change the Mashup ID " on page 1426

See also "Update Mashup or App Input Parameters" on page 1434.

Change the Web Part Title

You control whether the Web Part title is automatically generated or is editable with the **Override Web Part Title** option in the JackBe UI Settings category of the Web Part properties.

This option is set by default which causes the Web Part title to be generated automatically in the form *mashup-server* `Mashup:` *mashup-title* .

1. Clear this option to be able to enter your own title for the Web Part.
2. To change the title, once you have cleared this option, expand the **Appearance** category and enter a new title.
3. Then click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change the Mashup Server Connection for a Mashup Web Part

You can change which MashZone NextGen Server this Web Part connects to using these Web Part properties in theMashZone NextGenMashup category:

- **Mashup Server**: select another MashZone NextGen Server from the list of servers that are visible in the current site. When you select a MashZone NextGen Server, this also sets theMashZone NextGenURL,MashZone NextGenversion and SSO Application ID properties. This also determines whether SSO authentication (2010)/MOSS SSO (2007) or cookie forwarding are applicable.

  Note: The single sign on feature in MashZone NextGen Add-On for SharePoint works with the SharePoint Secure Service Store service (2010) or MOSS SSO (2007) to enable users to automatically log in to multiple MashZone NextGen Servers without having to sign in to each.

- **Override Mashup Server**: this option is not set by default. Set this option to override the current MashZone NextGen Server connection information.
- **MashZone NextGen Host**: this property is only accessible if you have set the override property. Change this to the URL for the MashZone NextGen Server this Web Part should connect to. Enter the URL in the form:

  `http://`*mashup-server-address-or-domain* `:`*port*

  Note: You cannot use the localhost domain name in MashZone NextGen Server URLs.

- **MashZone NextGen Version**: this property is only accessible if you have set the override property. Enter the MashZone NextGen version number in use in the MashZone NextGen Server that this Web Part should connect to.
- **SSO Application Name**: this property is only accessible if you have set the override property. Enter an SSO application name here to enable SSO support when connecting to this MashZone NextGen Server.

  SSO applications (2010)/MOSS SSO applications (2007) are configured by the SharePoint farm administrator. If SSO is not enabled, you and other users of the published page must enter MashZone NextGen credentials when viewing the Mashup Web Part.

Then click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change the Mashup ID

This task should only be performed by developers or advanced users. You can change which mashup displays in the Web Part by changing the **Mashup ID** property in theMashZone NextGenMashup category.

Note: Mashup IDs are not identical with mashup names.

Enter the MashZone NextGen ID of the mashup to display in this Web Part. This ID must exist in the MashZone NextGen Server defined in Web Part properties.

Then click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

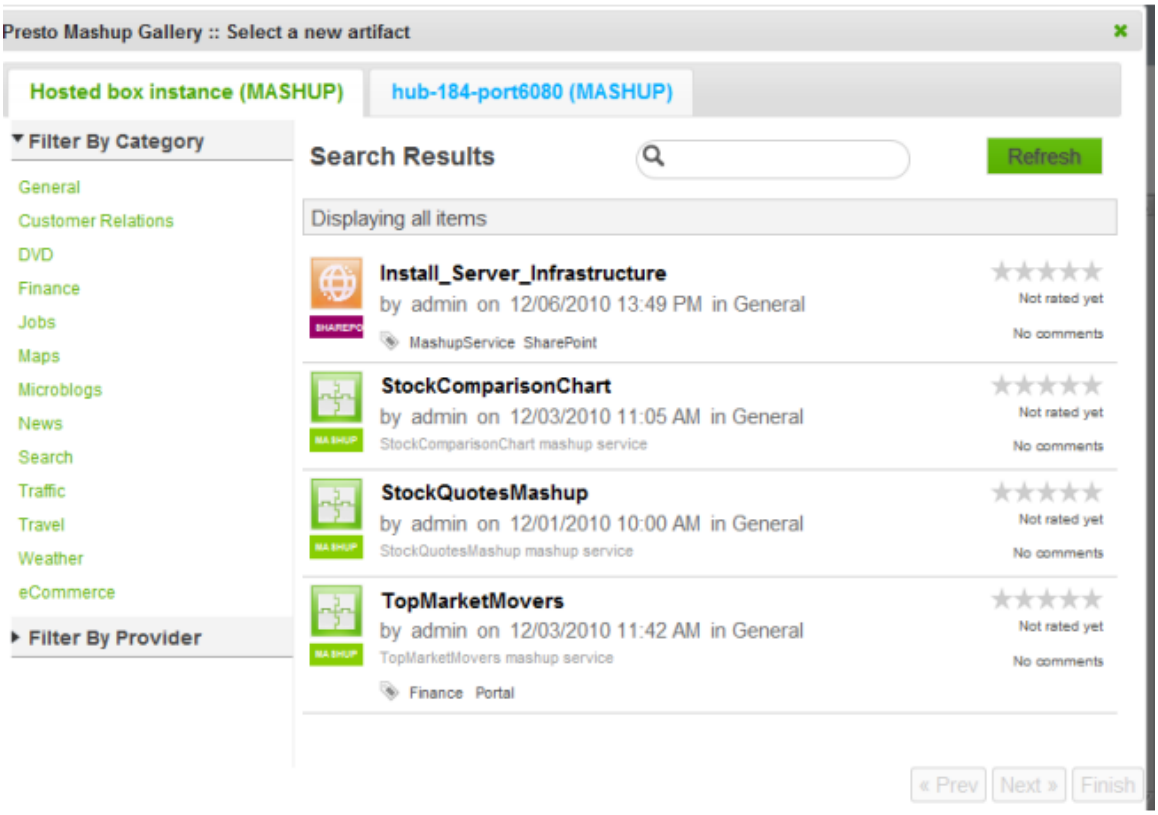
Publish Mashup Results as Native SharePoint Lists

You can create a copy mashup data as a native SharePoint list without using either Web Parts or External Content Types. This is best for mashups where results rarely or never change as the information is not updated in SharePoint once the list is created. The list does, however, fully participate in search or other integration techniques in SharePoint.

To publish a mashup directly as a native SharePoint list

1. In Site Actions, choose **More Options** (2010) or **Create** (2007).
2. In the Create dialog, choose **Lists >MashZone NextGenMashup List** and click **Create**.
3. Click **Select Mashup**, find the mashup you want in the MashZone NextGen Mashup Gallery and select it.

   The Mashup Gallery opens displaying a list of mashups. This list is cached and refreshes periodically. Use **Refresh** to manually refresh this list.



4. If the mashup has input parameters, update these parameters as needed. Click **Finish**.

Note: Review the input parameters values carefully as you cannot change this once you have selected the mashup.

5. Complete these properties for the List:
   - **List Title** defaults to the name of the mashup. Update this if needed.
   - Enter a **Description** for the List.
   - Choose one of the mashup fields as the **Title Field** if you want that column to be the title for each item in the List.
   - If prompted, enter the appropriate credentials to log into the MashZone NextGen Server for this mashup.
6. Click **Publish**.

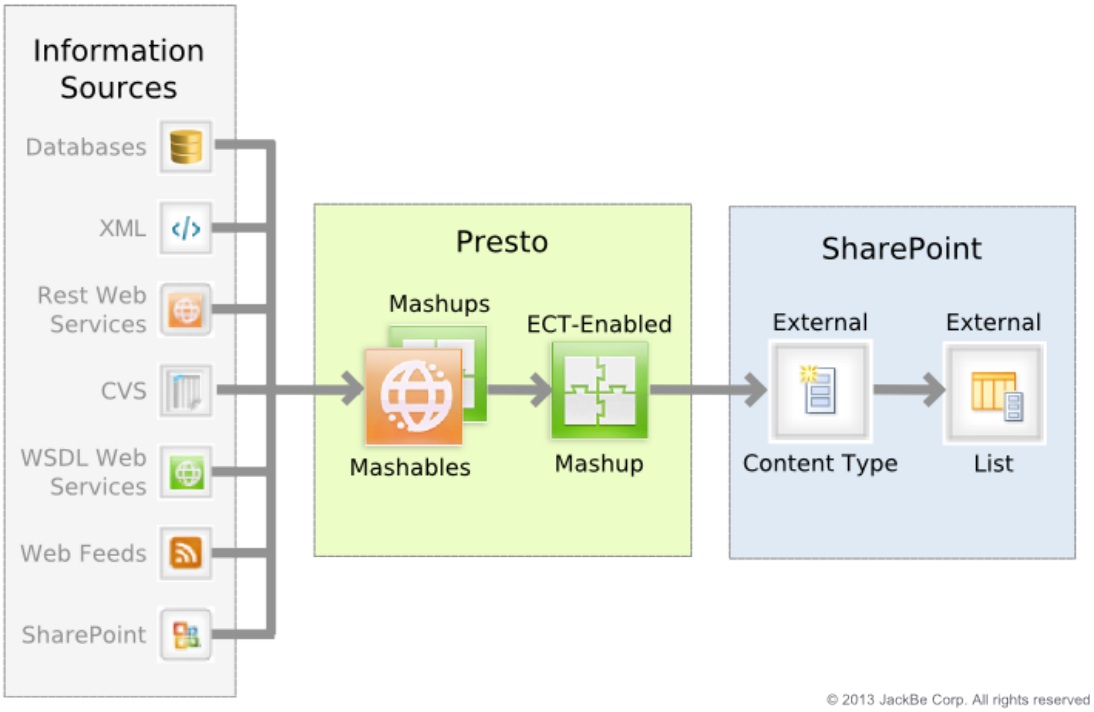
   The new list opens displaying each item from the mashup. This list is a copy of the mashup result when it is created. The list is not updated if mashup results change, nor can you update input parameters for the mashup, if any.

Publish Mashup Results as External SharePoint Lists

You can integrate information from many different external systems in your SharePoint 2010 sites using MashZone NextGen mashups as *external content lists*. If information of interest comes from a MashZone NextGen mashable, simply wrap the results in a compatible mashup in MashZone NextGen.

Important: This feature is *not* available for SharePoint 2007.

MashZone NextGen makes it simple to connect to many different types of information sources in your organization including databases, web services (using WSDLs or REST), web feeds (using RSS or Atom), CSV or XML. You can expose this data and integrate it in your SharePoint sites using MashZone NextGen mashups and SharePoint external content lists.



Consider the Requirements and Limitations and then follow The Mashup-to-External-List Process to get started.

Requirements and Limitations

You use MashZone NextGen Wires to create a mashup that SharePoint can access as an external content type. You use SharePoint Designer to define the external content type for mashups that are *ECT-enabled* and the external content lists based on these external content types.

Both MashZone NextGen and SharePoint must be configured properly to support communication between SharePoint and MashZone NextGen and between SharePoint Designer and MashZone NextGen. MashZone NextGen and SharePoint administrators must:

- *Enable one-way SSL* for clients connecting to the MashZone NextGenMashZone NextGen Server. This is required to allow SharePoint Designer to use a secure connection when you create external content types and external content lists based on MashZone NextGen mashups.

  MashZone NextGen administrators should see One-Way SSL to MashZone NextGen for instructions.

- *Use the Secure Service Store* in SharePoint to store credentials for MashZone NextGen used in connections from SharePoint and SharePoint Designer to these mashups.

  SharePoint administrators should see Configure Authentication for P4S 2010 for instructions on setting up SSO + Token Authentication.

In addition to basic connection requirements, there are specific requirements and limitations on the mashup results that you can use in external content lists and on the maintenance that is possible to these mashups. See "Supported SharePoint Operations" on page 315, "Data Requirements for the Results of ECT-Enabled Mashups" on page 316 and "Updates to the Mashup's Results" on page 317 for more information.

The Mashup-to-External-List Process

The steps to connect to an external system and integrate this dynamic information as using MashZone NextGen mashups are:

1. In MashZone NextGen Hub, create a mashable or mashup to connect to the external system with the information you want to use in a SharePoint list. See "Connect Information Sources as Mashables" on page 136 and "Creating Mashups in Wires" on page 284 for links and more information.
2. In Wires (in MashZone NextGen Hub), create a mashup that adapts the mashable or mashup you created in the previous step to allow SharePoint to use this mashup as an external content type. See "Adapt Results as a SharePoint External Content Type" on page 314 for instructions.
3. In SharePoint Designer, Create an External Content Type for a Mashup using the mashup created in the previous step.
4. In SharePoint Designer, Create an External Content List from a Mashup using the external content type created in the previous step to connect to the ECT-enabled mashup in MashZone NextGen.

   Save this list, add views or use any feature in SharePoint or SharePoint Designer to integrate this external list in your site.

Create an External Content Type for a Mashup

External content types define how to connect to a data source that is external to SharePoint and retrieve or perform other operations on specific information. It also defines metadata about the information itself, such as column name and datatypes.

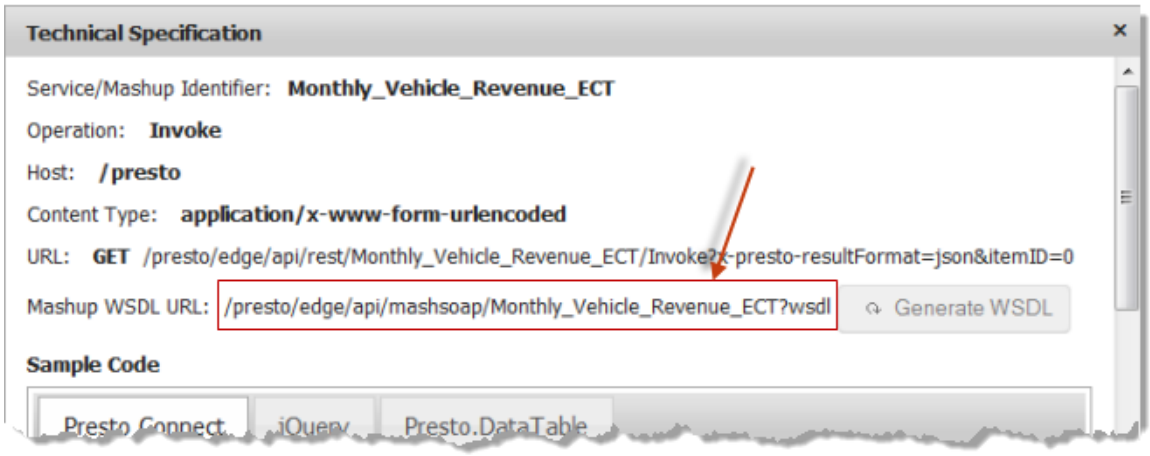
You use SharePoint Designer to customize your SharePoint sites, including defining external content types and creating external content lists. SharePoint Designer is a free tool that you download and install separate from SharePoint.

The basics of defining an external content type in SharePoint Designer are discussed in online help for SharePoint Designer, which covers how to create an external content type for a database. The process to create an external content type for mashups is slightly different:

1. Before you begin working in SharePoint Designer, open the ECT-enabled mashup that you want to work with in MashZone NextGen Hub. You must be the mashup owner or a MashZone NextGen administrator or developer to complete this step.

   a. Click **Show >** **Technical Spec** from the mashup's artifact page.

b. Find the Mashup WSDL URL and copy it. You will use this as you create an external content type for this mashup.



2. Open SharePoint Designer and open the site where you want to use mashup results.
   a. Select external content types and add a new content type.
   b. Enter a name for this external content type. Click the link for **External System**.
   c. Click **Add Connection**.
3. Choose `WCF Service` as the data source type.
4. Use the mashup WSDL URL you copied in step 1 as the **Service Metadata URL** and *change* the parameter at the end of the url to `sp_wsdl`.

   If the URL from the mashup technical spec is:

   ```
   https://myOrg.mservers.com:8443/presto/edge/api/rest/
   Monthly_Vehicle_Revenue_ECT?wsdl
   ```

   Then change this to:

   ```
   https://myOrg.mservers.com:8443/presto/edge/api/rest/
   Monthly_Vehicle_Revenue_ECT?sp_wsdl
   ```

5. Use the mashup WSDL URL you copied in step 1 as the **Service Endpoint URL** and *remove* the `?wsdl` parameter from the URL. With the WSDL URL shown in the previous step, the service endpoint would be:

   ```
   https://myOrg.mservers.com:8443/presto/edge/api/rest/
   Monthly_Vehicle_Revenue_ECT
   ```

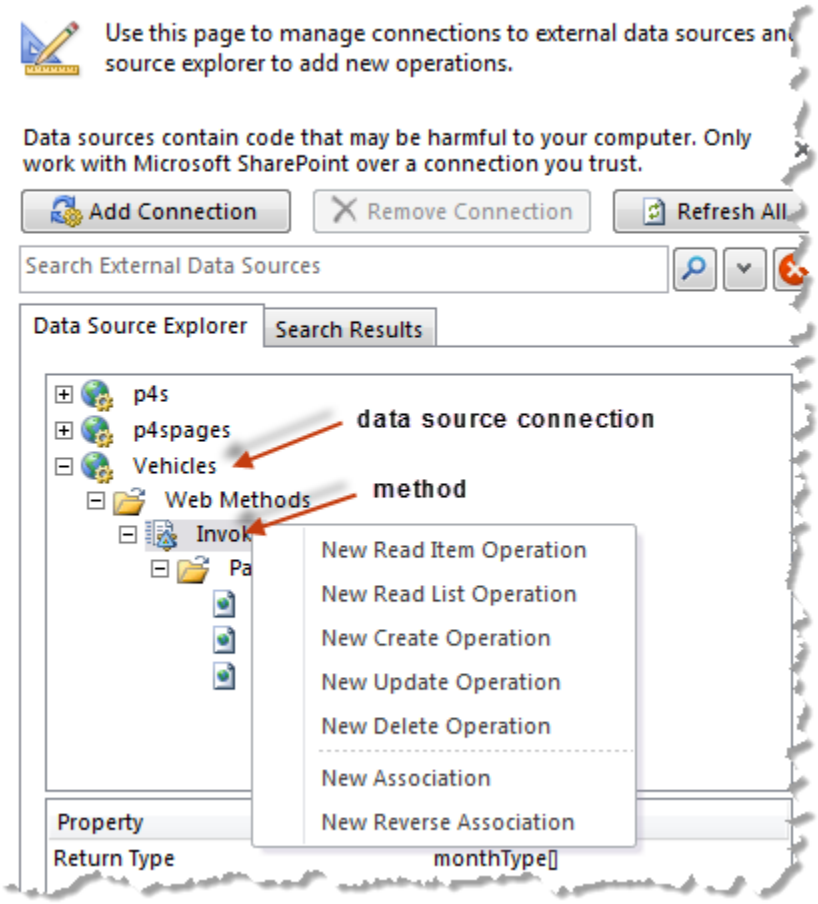
6. Change the WCF Service Authentication Settings to **Connect with Impersonated Custom Identity**.
7. Enter the name of the **Secure Service Application ID** for the MashZone NextGen Server that hosts this mashup.

   Contact your SharePoint administrator to get the correct application ID for this field.

8. Update any other settings and click **OK**.

> **Note:** In some cases, you may get an error loading URLs as SharePoint validates both the WSDL and endpoint URLs. See "Troubleshooting URL Loading Problems" on page 1433 for suggestions.

9. The connection you created should now be listed. Expand this till you see the `Invoke` method.



10. Right click the method and create a **New Read Item Operation**.

    This opens a wizard to define this operation. You can change names or display names and other properties as needed. You *must* map both the input parameter and the return parameter used as the item ID to the Identifier for lists of this type.

11. Right click the method again and create a **New Read List Operation**.

12. Once these two operations are defined, save this external content type.

You can now add external content lists based on this type.

Create an External Content List from a Mashup

Once you have an external content type defined for an ECT-enabled mashup, you can add external content lists with mashup results to SharePoint sites:

1. In SharePoint Designer, open the site you want to work in and select External Content Types.
2. Select the external content type for the mashup you want to use and click **Create Lists and Forms** in the ribbon.
3. Give the list a name, choose the operation to get an individual list item and save this list.

   No forms are needed unless the mashup supports create, update or delete operations.

Troubleshooting URL Loading Problems

When you create new connections, you may receive errors trying to load the WSDL URL or the endpoint URL. The most common reasons for these errors are:

- Minor typos in the URLs.
  - Make sure the URL for the WSDL ends in `?sp_wsdl`.
  - Make sure the endpoint URL has *no* parameters (`?sp_wsdl`) at the end.
- The wrong Secure Service Application ID.

  This application stores MashZone NextGen credentials that are mapped to different SharePoint users. Check with your SharePoint administrator and verify that:
  - The application ID is correct.
  - Your MashZone NextGen credentials exist in this application and are correct.
  - This MashZone NextGen account has permission to use this mashup.
- A missing certificate for this MashZone NextGen Server.

  SharePoint Designer uses a secure connection to the MashZone NextGen Server. You must install the digital certificate for this MashZone NextGen Server on your computer to connect successfully.

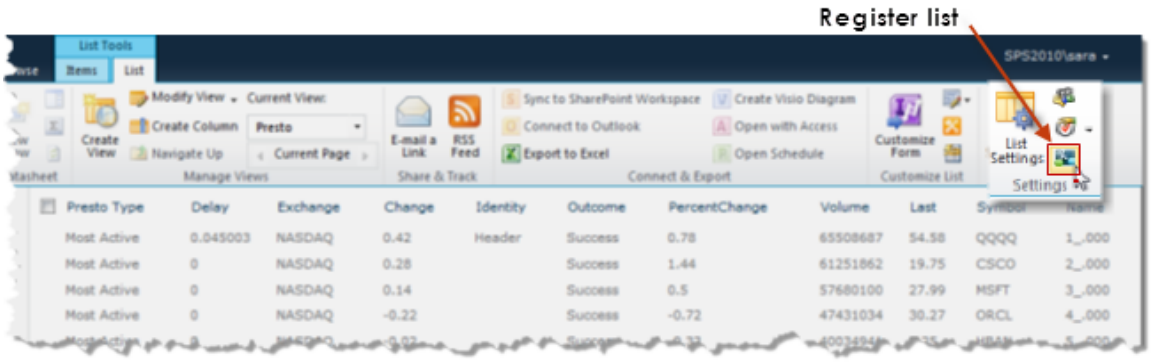
  1. Contact your MashZone NextGen administrator to obtain the correct certificate. Typically, this is a public.cer file.
  2. Double click this file to start the wizard to install this certificate.
  3. Do *not* install the certificate in the default or personal certificate store. Set the option to choose the certificate store.
  4. Select `Trusted Root Certificate Authorities` as the certificate store.

Register a SharePoint List as a MashZone NextGen Mashable

You can use information from SharePoint Lists in MashZone NextGen mashups or apps by registering Lists as a *mashable information source*.

To register a List in MashZone NextGen

1. In SharePoint 2010, open a list and click **List** in the List Tools tab.
2. In SharePoint 2007, open a list and click *Actions*.
3. Select **Register List** in the ribbon toolbar.



4. If desired, change the **Name** or **Description** to assign to theMashZone NextGenmashable. The name defaults to the List title.
5. If this List has views, choose the specific view you want to use for this MashZone NextGen mashable.
6. In **Publish To**, select the MashZone NextGen Server where you want to register this List.
7. If desired, change **SharePoint Access Credentials** to control which user credentials are used in MashZone NextGen to retrieve information from this List:
   - **Bind now** = set this option to always use the credentials you have supplied while registering this SharePoint list as a mashable.
   - **Bind at runtime dynamically** = this is set by default. With this option, MashZone NextGen uses the credentials of the current user to access information for this SharePoint List, rather than your credentials.
8. Click *Submit* to reigster the List as a mashable.

Update Mashup or App Input Parameters

Input parameters allow mashups and apps to produce and display information that is tailored to your specific needs. For example, a mashup to find businesses within a given location might have input parameters for the type of business to find and for the city and state to use as the location. Input parameters may also determine how to filter information or how to sort it, among many other functions.

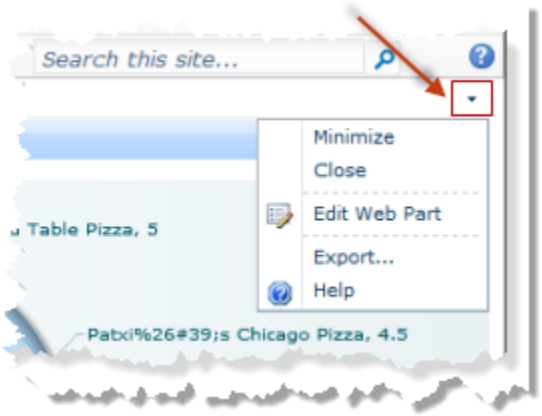
Not all mashups or apps have input parameters, not all input parameters are required. You can update input parameters for mashups or apps in:

- The Mashup Gallery or App Gallery when you publish mashups or apps to Web Parts.
- The app itself, from a toolbar button in the app title bar.

  This title bar is hidden, by default, but you can update the Web Part to show this instead. See "Hide or Show App or View Titles and Toolbars" on page 1443 for more information. If this title bar is visible, any user who has permissions to use the app can update input parameters.
- The Web Part properties pane for the Mashup or App Web Part.

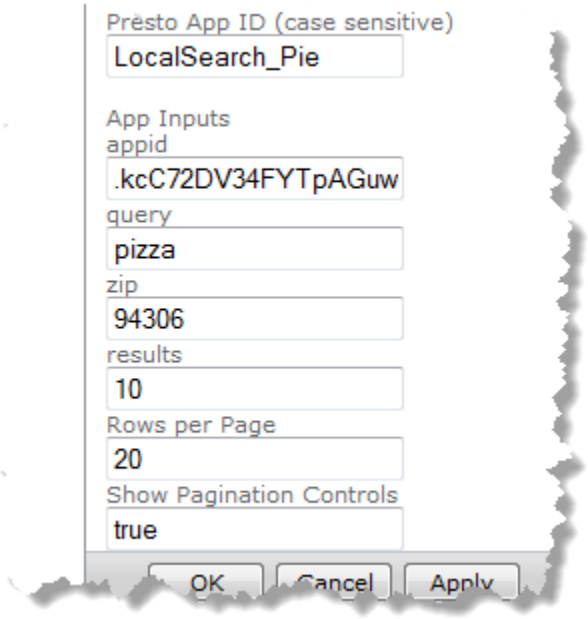
To edit input parameters from Web Part properties

1. Click the down arrow in the right corner of the Mashup or App Web Part title bar and select **Edit Web Part**.



   This puts the page in Edit Mode and opens the Web Part Properties pane for this mashup or app.

2. Find the input parameter fields in the **MashZone NextGen** or **MashZone NextGen** category.

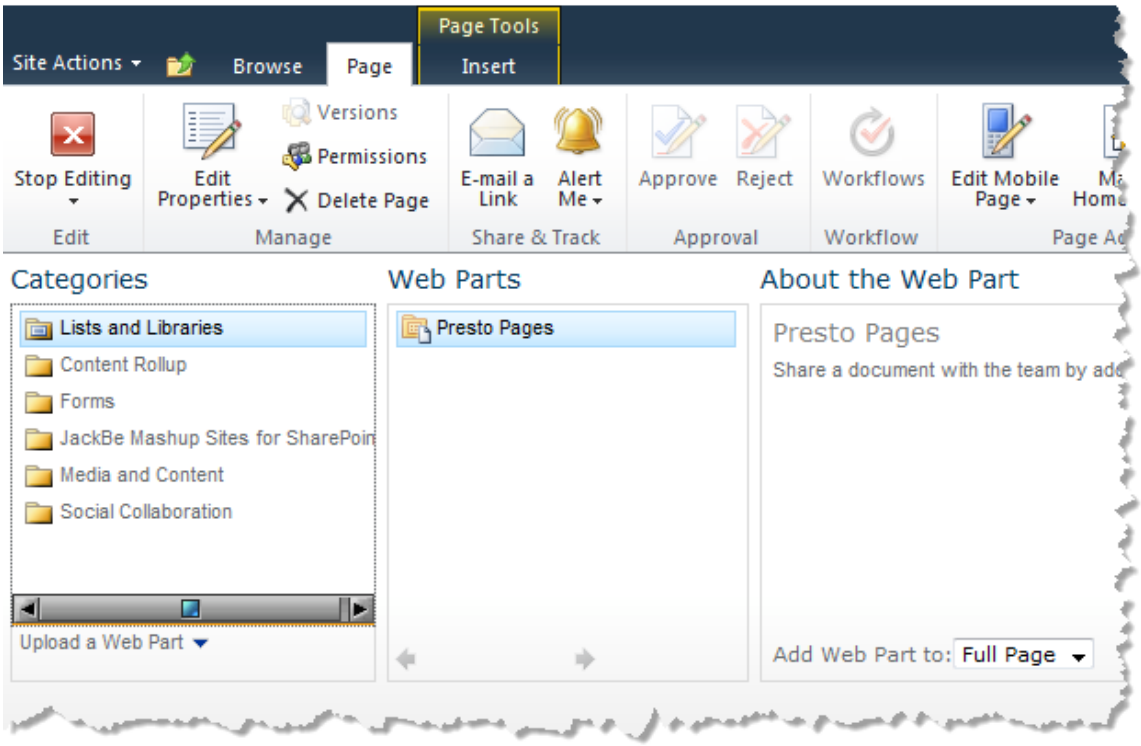


3. Update the input parameter values as needed.
4. For some apps, you can also Change Pagination or Themes.
5. Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

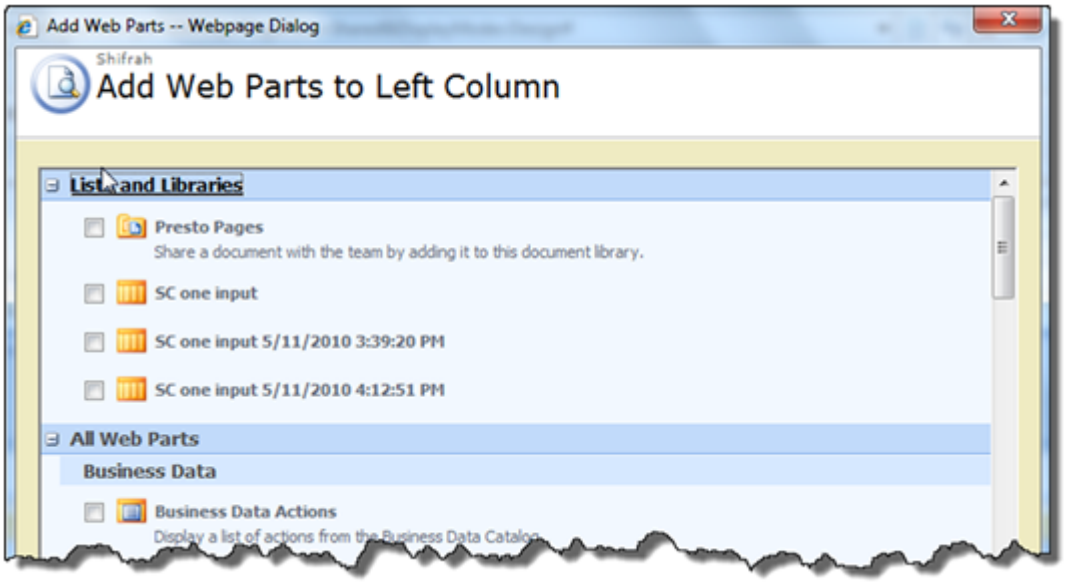
Add an App Web Part to the Web Part Page

In order to publish apps to a Web Part Page, you must add one or more app Web Parts to the zones of that page. You then publish apps to each App Web Part.

1. In SharePoint 2010, click **Add a Web Part** in the zone on your Web Part page where you would like to place an app. The page opens in edit mode with a list of categories to choose from:



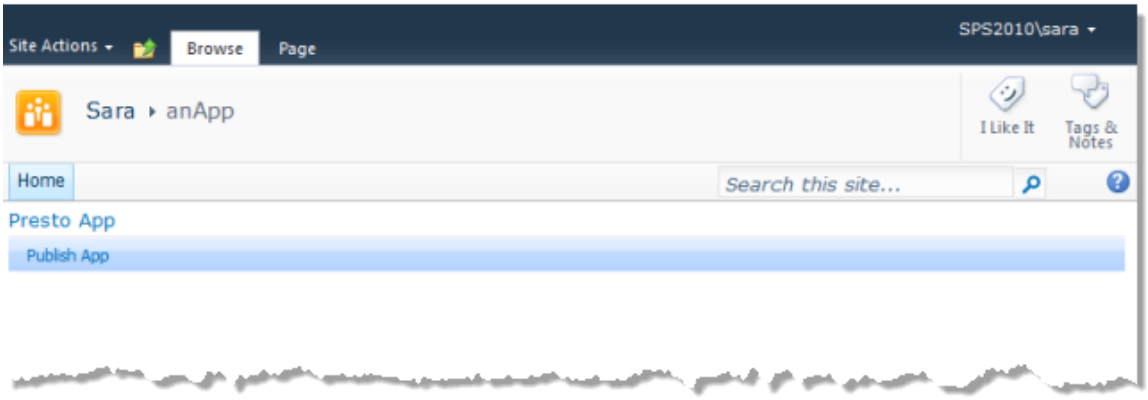
2. In SharePoint 2007, Click **Add a Web Part** in the zone on your Web Part page where you would like to place an app. The Add Web Part menu opens.



3. Select the **JackBe Mashup Sites for SharePoint** category and set the **App Web Part** option.
4. Click **Add** (in the bottom right corner).
5. Continue editing this Web Parts page. You can:

- Add appropriate Web Parts to other zones on the page.
- Edit the properties for your App Web Part, or any Web Part on the page. See "Update App Web Part Properties" on page 1441 for more information.
- Select and drag Web Parts to move them to other zones.

6. Once you are finished adding and modifying Web Parts, click **Stop Editing** in the Web Part page toolbar. The page returns to the normal display mode, with an App Web Part and toolbar in the zones you have selected.



You can now Publish an App to an App Web Part.

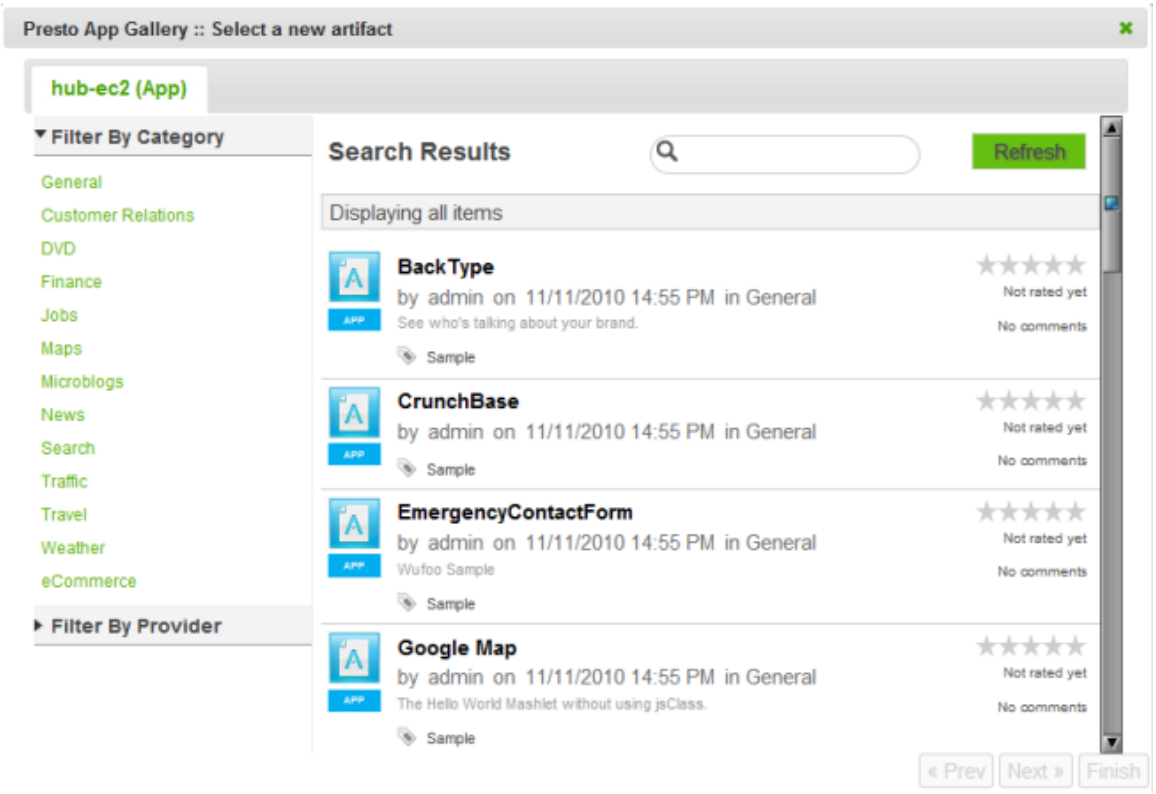
Publish an App to an App Web Part

Once you have stopped editing your Web Part page, you can select the specific app to use in any App Web Parts on your page.

1. Click **Publish App** from the toolbar in any of the App Web Parts on your page.
2. If prompted, enter credentials to log in to the MashZone NextGen Servers you want to work with. You can also **Skip** specific MashZone NextGen Servers. Apps from the MashZone NextGen Servers that you skip will not be listed.

   Note: Do *not* include the Windows domain in your username, such as `mydomain\myname`, when you log in. The MashZone NextGen Server does not accept domain names. You may also need to add MashZone NextGen Servers to your trusted sites or change your cookie preferences to allow session cookies. If you still receive login errors, check with your MashZone NextGen administrator.

   The App Gallery opens displaying a list of apps. This list is cached and refreshes periodically. Use **Refresh** to manually refresh this list.



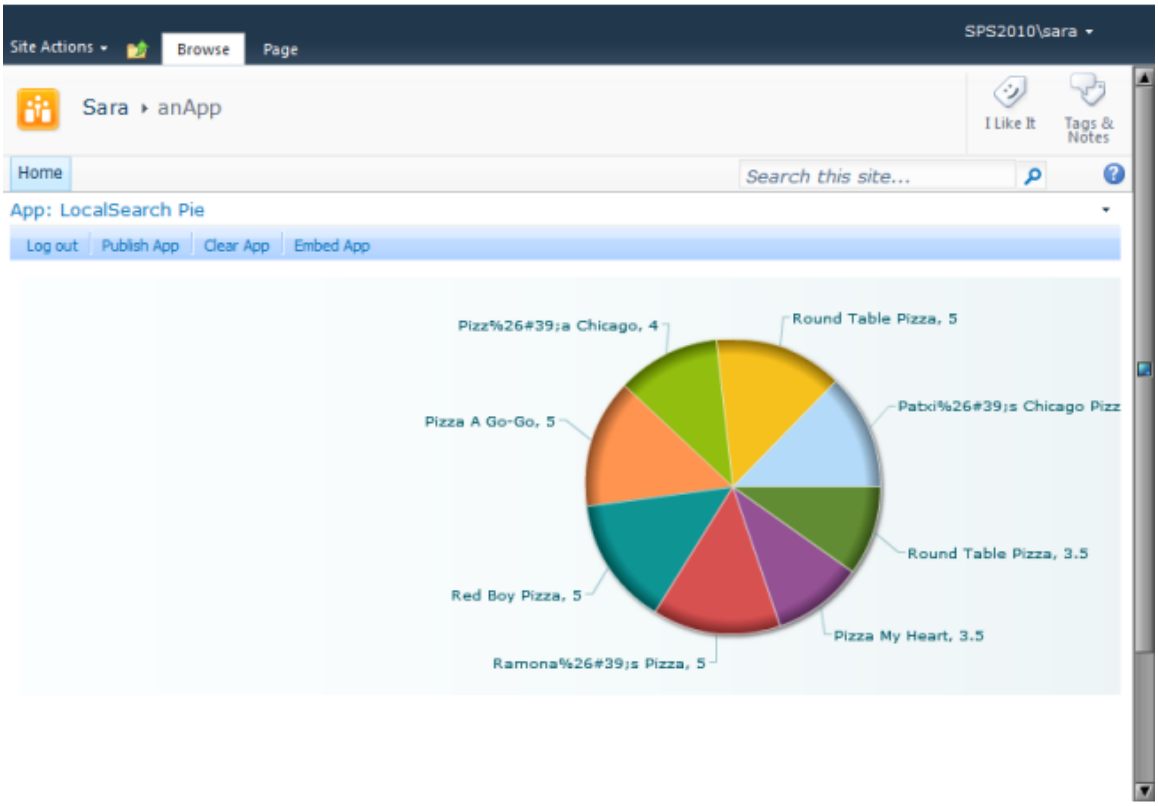
3. Find and select the app you want to add to this Web Part.

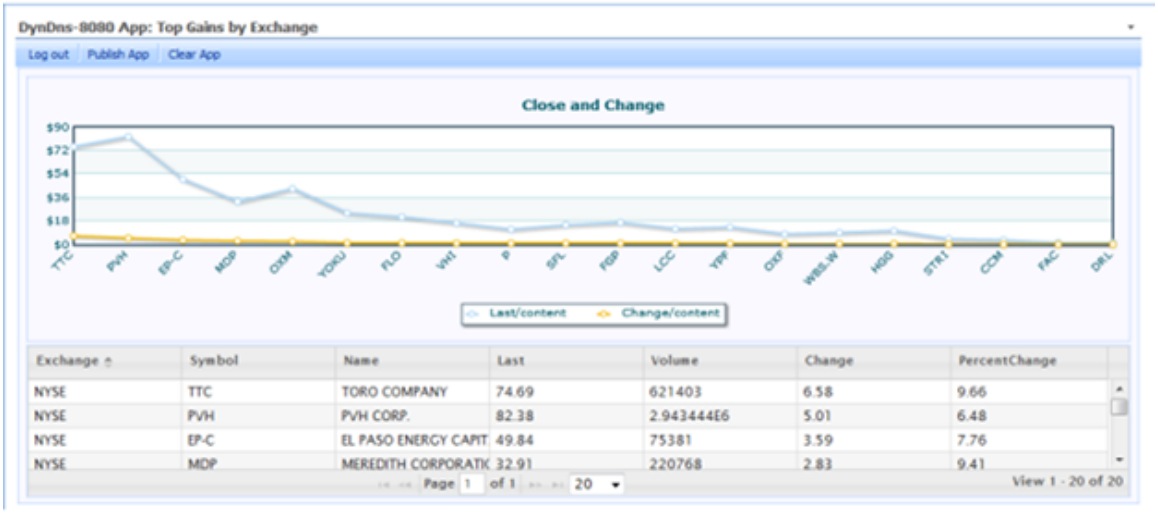
   You can filter the list of available apps by category, by provider or enter a partial name in the search field.

4. If the app has input parameters, update these parameters as needed.

5. Click **Finish**.

   The app you selected displays in the zone for this Web Part.

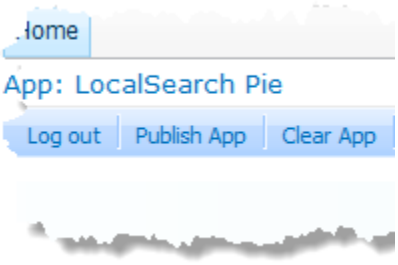




If you have appropriate permissions, this includes a toolbar with actions for this app. See "App Web Part Toolbar" on page 1441 for more information. See also "Update App Web Part Properties" on page 1441 for other tasks you can perform.

App Web Part Toolbar

If you have permission to edit a Web Part Page, the app displays a toolbar with editing actions. The actions that appear in the toolbar change based on the status of the Web Part.



Actions on this toolbar include:

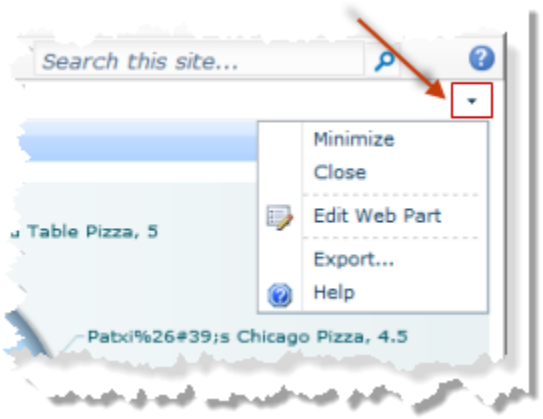
- **Log Out**: ends your session with the MashZone NextGen Server for this app. You can then log back in using different credentials. Simply click **Log Out**, enter another username and password and click **Log In**.
- **Publish App**: allows you to find and add a different app to this App Web Part. See "Publish an App to an App Web Part" on page 1438 for instructions.
- **Clear App** removes the current app in this App Web Part. Use **Publish App** to find and add a different mashup.

In SharePoint 2007, you may hide or show the App Web Part Toolbar with **Site Actions >Hide Web Part Menu Bar** or **Show Web Part Menu Bar**.

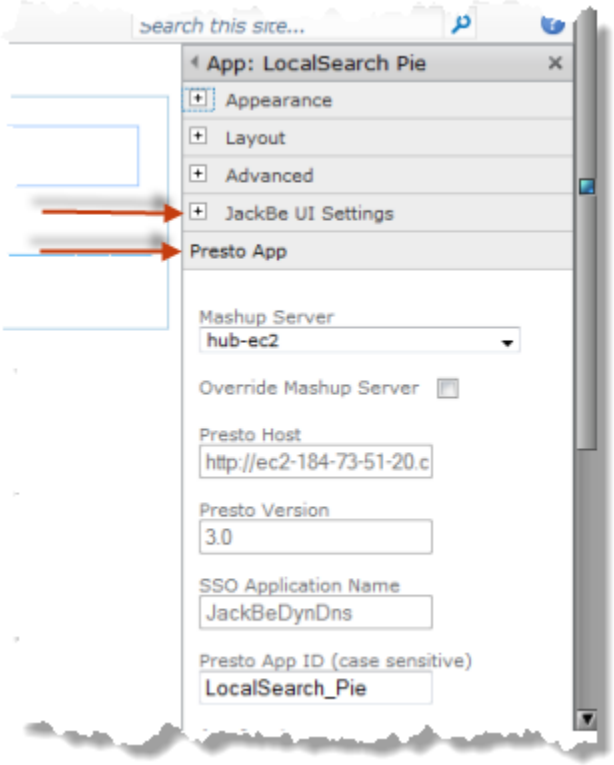
Update App Web Part Properties

You can update Web Part properties to make changes to the appearance, layout and advanced features of an App Web Part. Changes you make to the Web Part properties are visible to anyone viewing the published Web Part Page.

To edit App Web Part properties, click the down arrow in the right corner of the App Web Part title bar and select Edit Web Part.



This puts the page in Edit Mode and opens the Web Part Properties pane for this app:



In addition to the standard Web Part properties, app Web Parts have two additional categories of properties that you can update: JackBe UI Settings and MashZone NextGen App. Use these properties to:

- Change the Web Part Title
- Hide or Show App or View Titles and Toolbars
- Change Pagination or Themes
- Enable Anonymous Access

- Change the Mashup Server Connection for an App Web Part
- Change the App ID

See also "Update Mashup or App Input Parameters" on page 1434.

Change the Web Part Title

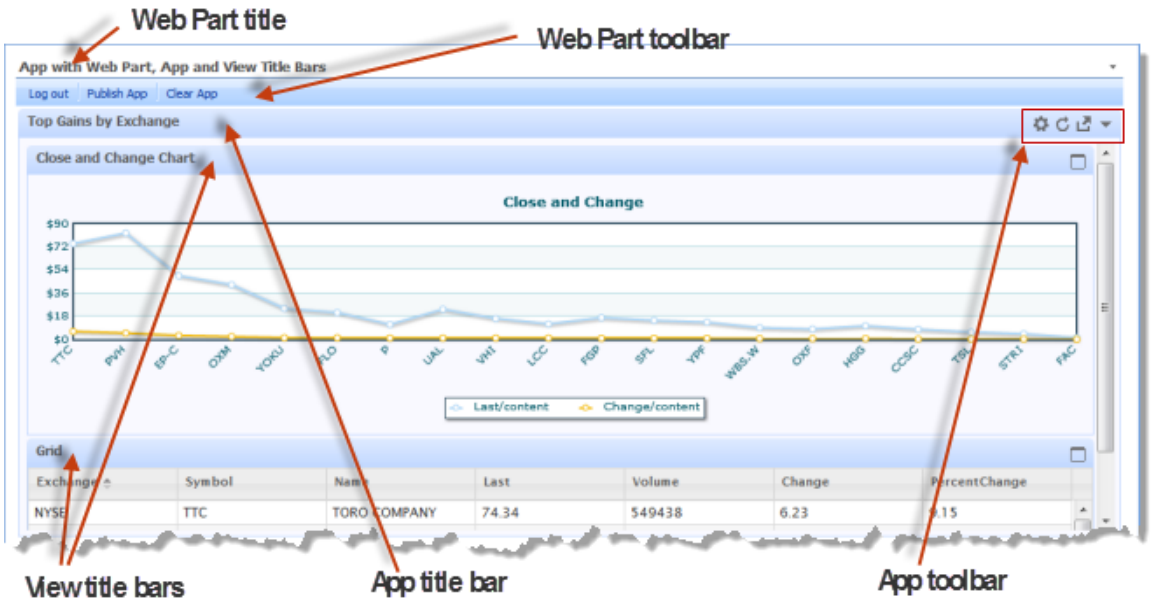
You control whether the Web Part title is automatically generated or is editable with the **Override Web Part Title** option in the JackBe UI Settings category of the Web Part properties.

This option is set by default which causes the Web Part title to be generated automatically in the form *mashup-server* `App:` *app-title* . To change this default title:

1. Clear the **Override Web Part Title** option in the JackBe UI Settings category.
2. Expand the **Appearance** category and enter a new title.
3. Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Hide or Show App or View Titles and Toolbars

Both the App Title Bar and the View Title Bars contain titles and toolbars with buttons to assist users, as shown in this example:



These title bars do take additional space, but also can provide useful functionality. The toolbar in the App Title Bar allows users to update input parameters at any time for apps that have them. It also allows uers to refresh the app at any time. The toolbar in View Titles Bars allows users to maximize a specific view to see it in more detail or show all views (the default).

By default both the App Title Bar and the View Title Bars are hidden when you publish an app in Web Part. You can show or hide each of these title bars using these two Web Part properties in the JackBe UI category:

- **Disable Header**: hides or shows the App Title and toolbar. Clear this option to show this title and toolbar.
- **Disable View Header**: hides or shows the View Titles and toolbars for apps that have multiple view. Clear this option to show these titles and toolbars.

Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change Pagination or Themes

Some apps have the following properties that determine how much data is shown in the app and the color theme used for borders and title bars in the app:

- **Enable pagination**: For apps with lots of data. showing smaller subsets of data provides better performance and can make it easier to find specific data of interest. If this property is set, the app displays just one 'page' of data. It also includes a toolbar to allow users to move to other pages of data.
- **Rows Per Page**: This only applies if pagination is enabled. This determines how many rows of data are included in one page.
- **Theme**: Defines a color scheme for borders for the app and, if shown, for the App and View Title bars. See "Hide or Show App or View Titles and Toolbars" on page 1443 for more information.

Enable Anonymous Access

Anonymous access allows guest users who are not logged in to view and work with an app. You only need to set the *Enable Anonymous Access* option for an App Web Part if guest users should have access and neither SSO nor cookie forwarding is in use with this site to automatically provide user credentials.

Change the Mashup Server Connection for an App Web Part

You can change which MashZone NextGen Server this Web Part connects to using these Web Part properties in the MashZone NextGen App category:

- **Mashup Server**: select another MashZone NextGen Server from the list of servers that are visible in the current site. When you select a MashZone NextGen Server, this also sets the MashZone NextGen URL, MashZone NextGen version and SSO Application ID properties. This also determines whether SSO authentication and cookie forwarding are applicable.

  Note: The single sign on feature in MashZone NextGen Add-On for SharePoint works with the SharePoint Secure Store service to enable users to automatically log in to multiple MashZone NextGen Servers without having to sign in to each.

- **Override Mashup Server**: this option is not set by default. Set this option to override the current MashZone NextGen Server connection information.
- **MashZone NextGen Host**: this property is only accessible if you have set the override property. Change this to the URL for the MashZone NextGen Server this Web Part should connect to. Enter the URL in the form:

  `http://`*mashup-server-address-or-domain* `:`*port*

  > **Note:** You cannot use the localhost domain name in MashZone NextGen Server URLs.

- **MashZone NextGen Version**: this property is only accessible if you have set the override property. Enter the MashZone NextGen version number in use in the MashZone NextGen Server that this Web Part should connect to.
- **SSO Application Name**: this property is only accessible if you have set the override property. Enter an SSO application name here to enable SSO support when connecting to this MashZone NextGen Server.

  SSO applications are configured by the SharePoint farm administrator. If SSO is not enabled, you and other users of the published page must enter MashZone NextGen credentials when viewing the Mashup Web Part.

Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change the App ID

This task should only be performed by developers or advanced users. You can change which app displays in the Web Part by changing the **App ID** property in the MashZone NextGen App category.

> **Note:** App IDs are not identical to app names.

Enter the MashZone NextGen ID of the app to display in this Web Part. This ID must exist in the MashZone NextGen Server defined in Web Part properties.

Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

MashZone NextGen Add-On for SharePoint User Guide

Mashups and apps enable your sites in SharePoint to display dynamic, up-to-the-minute, integrated data that is specifically relevant to your organization, suppliers, customers or other partners. SharePoint Web Part Pages allow you to construct dashboards from many different types of content to give viewers a coherent and multi-faceted source of information.

Users easily create mashups and apps using simple, graphical tools in MashZone NextGen that build on many different *information sources* such as web services, web

feeds, databases or existing systems in your organization.The MashZone NextGen Add-On for SharePoint allows you to easily incorporate these mashups and apps as native Web Parts in your SharePoint sites, providing deep integration with other SharePoint content using SharePoint's Web Part communication framework. For mashups, you have additional integration options to publish them directly as native SharePoint Lists or integrate them dynamically as external Lists using the Business Connectivity Service (BCS).

- *Mashups* provide a powerful way to gather relevant data for research and analysis. You have three choices on how to publish mashups that affect the features available in SharePoint:

Mashup Published	Shows Dynamic Data	Users Can Change Input Parameters or Properties	Accessible to SharePoint Services
In Web Parts Mashups published in Web Parts allow you to use mashup data on a Web Part Page in an easily readable table format, similar to a list view. See "Publishing Apps and Mashups to Web Parts" on page 1416.	✓	✓	
As native SharePoint Lists See "Publish Mashup Results as Native SharePoint Lists" on page 1427.			✓
As SharePoint External Lists See "Publish Mashup Results as External SharePoint Lists" on page 1428.	✓		✓

With Web Parts or External Lists, changes in mashup results are visible in real-time in SharePoint. With native Lists, the data is just a snapshot of mashup results when the mashup was published.

With Web Parts, users that have edit permissions have added flexibility to change input parameters for the mashup and see different data. They can also edit Web Part properties for the mashup.

With native or external Lists, data within the mashup results is handled just like any SharePoint List. Users can search through the data or use this with other SharePoint services, such as Performance Point.

- *Apps* combine mashups or other information sources with one or more *views* of that information including charts, tables and maps. Different views can convey information in more accessible, effective, and memorable formats.

  You can also use apps to connect to a remote web service to view, update, add, or remove data. For instance, you could input travel expenses in a simplified form.

  Apps can also be combined to work together in dashboards or workspaces. Apps are also easily shared with others in the MashZone NextGen Enterprise AppDepot, Enterprise Java Portals, SharePoint or any web page.

  See "Publishing Apps and Mashups to Web Parts" on page 1416.

Publishing Apps and Mashups to Web Parts

The basic tasks you complete to add apps or mashups to SharePoint pages using Web Parts are:

- Create a Web Part Page.
- Add a Mashup Web Part to a Web Part Page or Add an App Web Part to the Web Part Page .
- Publish a Mashup to a Mashup Web Part or Publish an App to an App Web Part

Other Tasks

For mashups, you can Publish Mashup Results as Native SharePoint Lists or Publish Mashup Results as External SharePoint Lists instead.

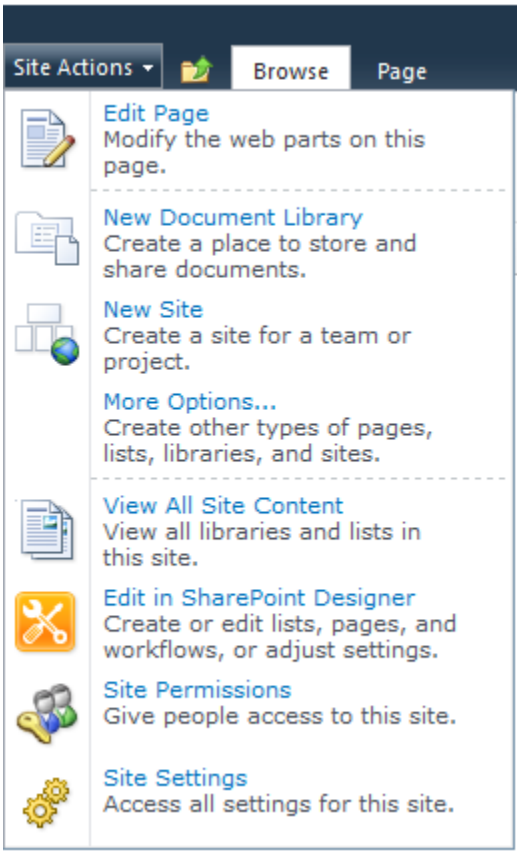
In addition to adding mashups or apps to Web Part Pages in SharePoint, you can also use information in SharePoint lists in MashZone NextGen as a mashable information. To use list information in MashZone NextGen, see "Register a SharePoint List as a MashZone NextGen Mashable" on page 1434.

Create a Web Part Page

You control and customize the layout and content of a Web Part page. A page is composed of one or more Web Parts that may contain apps, mashups or other information from inside and outside of SharePoint.

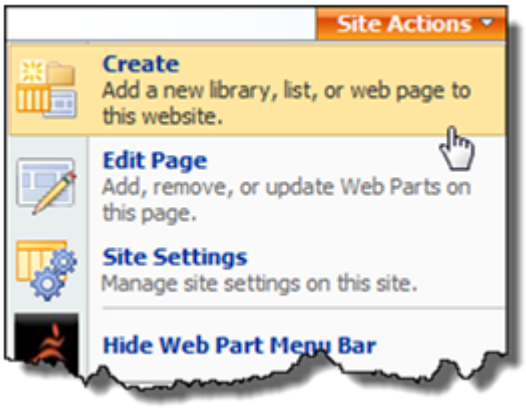
To create a Web Part page

1. In SP 2010, click **Site Actions > More Options.** The Browse Actions window opens.



Then filter actions by **Page**, choose **Web Part Page** and click **Create**.

2. In SP 2007, click **Site Actions > Create**.



Then select **Web Part Page** under Web Pages.

3. Name your Web Part Page and set the Overwrite option if this page should overwrite an existing page with the same name in the document library. This is the file name that appears in headings and links throughout the site.

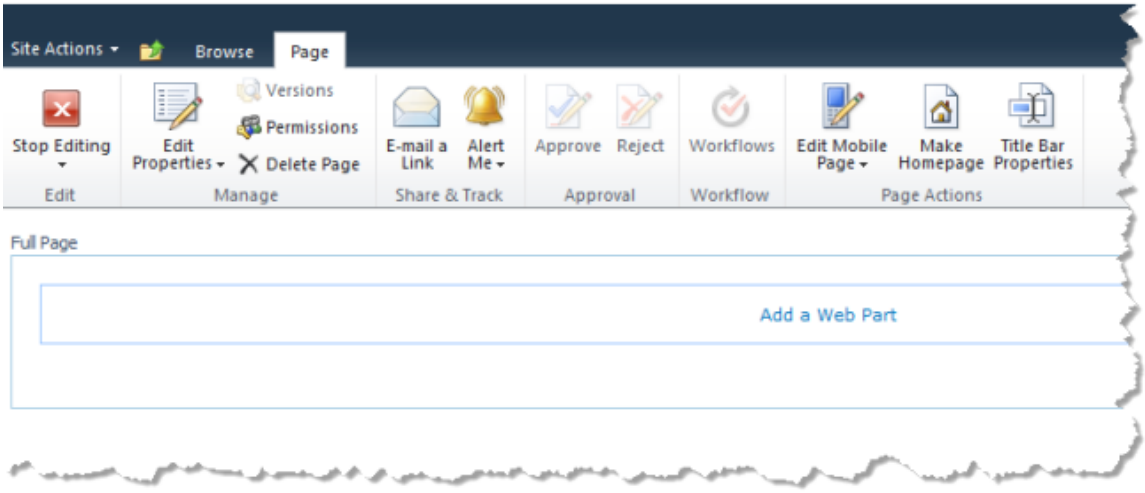
4. Select a layout from the **Choose a Layout Template** list. A preview of the selected layout appears to the left.

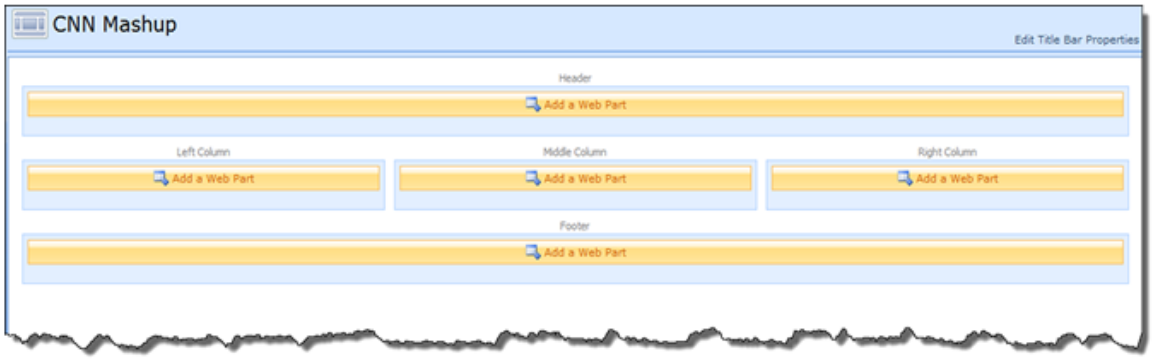
5. Choose the **Document Library** where you want the Web Part Page to be saved.

> **Important:** Do not save Web Part pages in a Document Library that is configured to require checkout before modifications. You cannot add mashups or apps to Web Part Pages in this case.

6. Click Create.

   The new Web Part page opens in Edit Mode with zones displayed, based on the layout you selected, where you can add Web Parts.





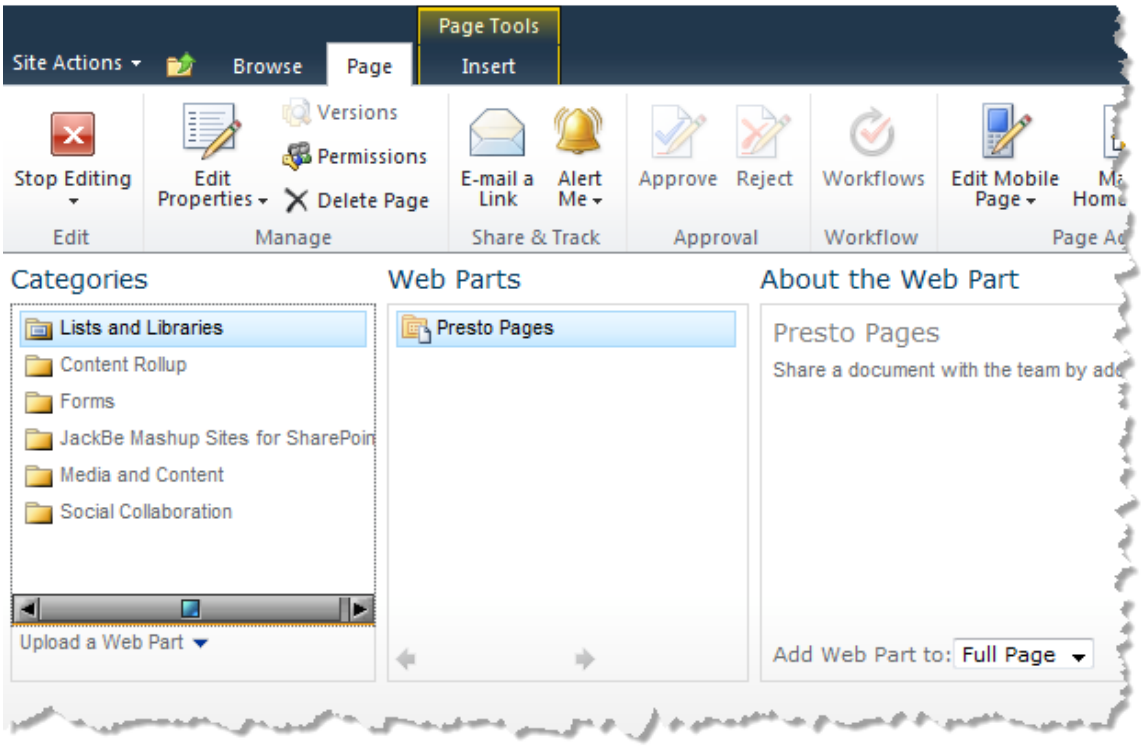
In Edit Mode, you can use the toolbar that appears above the page layout to edit page properties, delete the page, edit title properties or perform other actions. Use **Edit Title Bar Properties** to change the page title or add a caption, description, or image.

You can now Add a Mashup Web Part to a Web Part Page or Add an App Web Part to the Web Part Page and then publish mashups or apps to these Web Parts.

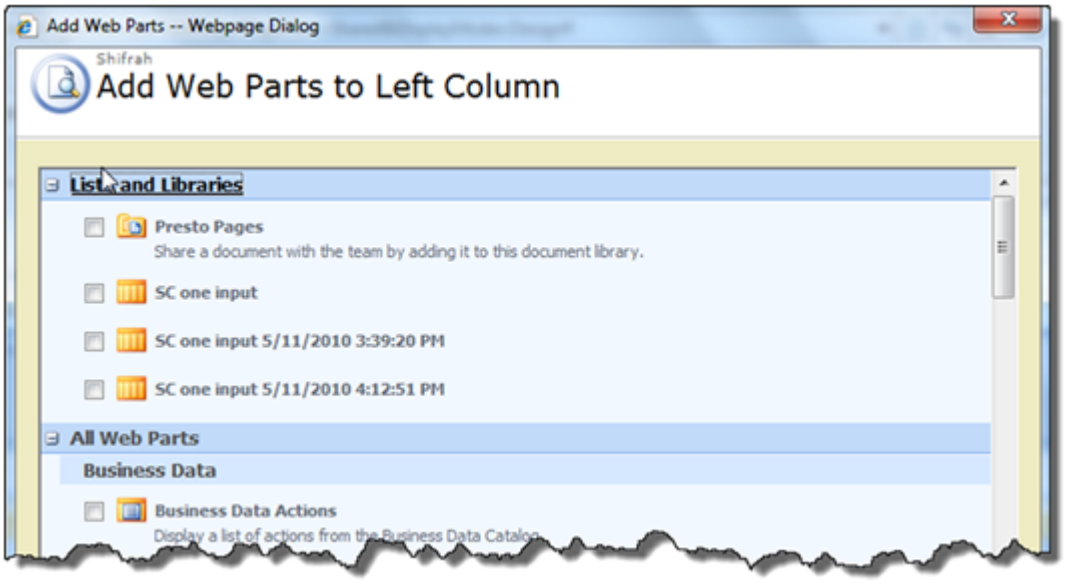
Add a Mashup Web Part to a Web Part Page

In order to publish a mashup to a Web Part Page, you must add a Mashup Web Parts to a zone in that page. You then publish mashups to each Mashup Web Part to display data from those mashups in table format.

1. If the page where you want to place a mashup is not already in edit mode, edit the page.
2. Click **Add a Web Part** in the zone on your Web Part page where you would like to place a mashup. In SP 2010, Web Part filters open:



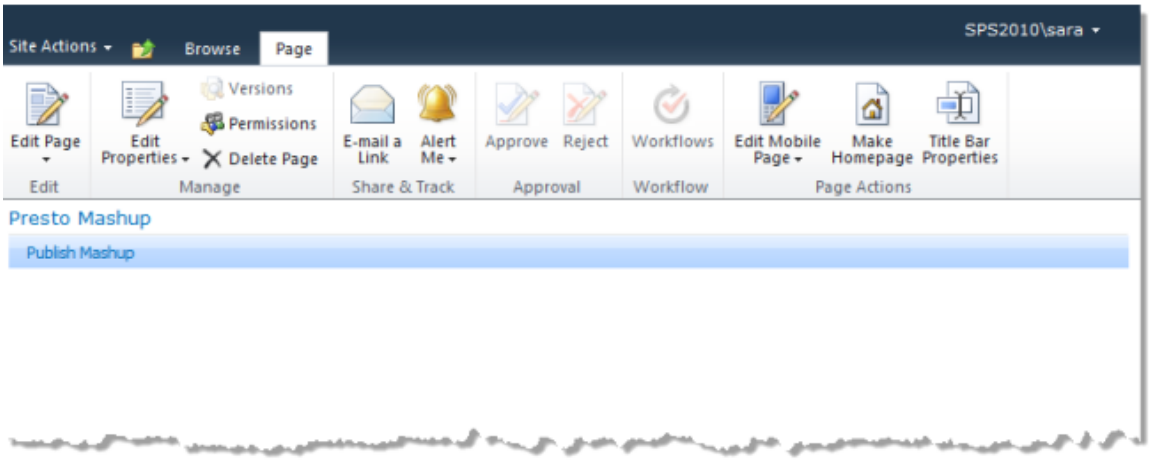
In SP 2007, the Add Web Parts menu opens:



3. Select the **JackBe Mashup Sites for SharePoint** category and click **Mashup Web Part**.
4. Click **Add** (in the bottom right corner).
5. Continue editing this Web Parts page, as needed. You can:
   - Add appropriate Web Parts to other zones on the page.

- Edit the properties for your Mashup Web Part or for any Web Part on the page. See "Update Mashup Web Part Properties" on page 1424 for more information.
- Select and drag Web Parts to move them to other zones.

6. Once you are finished adding and modifying Web Parts, click **Stop Editing** in the Web Part page ribbon. The page returns to the normal display mode, with a Mashup Web Part and toolbar in the zones you have selected.



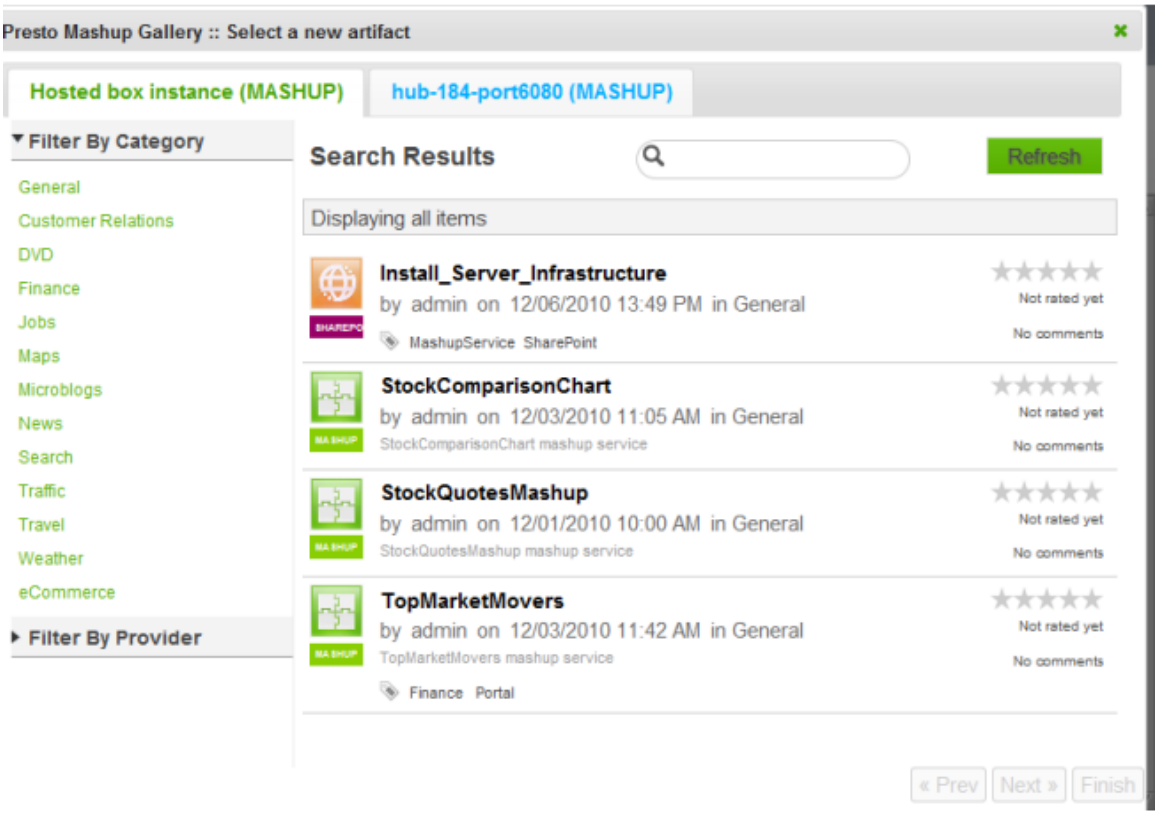
Publish a Mashup to a Mashup Web Part

Once you have stopped editing your Web Part page, you can select the specific mashup to use in any Mashup Web Part on your page from the Mashup Gallery.

1. Click **Publish Mashup** from the toolbar in any of the Mashup Web Parts on your page.
2. If prompted, enter credentials to log in to the MashZone NextGen Servers you want to work with. You can also **Skip** specific MashZone NextGen Servers. Mashups from the MashZone NextGen Servers that you skip will not be listed.

> **Note:** Do *not* include the Windows domain in your username, such as `mydomain\myname`, when you log in. The MashZone NextGen Server does not accept domain names. You may also need to add MashZone NextGen Servers to your list of trusted sites or change your cookie preferences to allow session cookies. If you still receive login errors, check with your MashZone NextGen administrator.

The Mashup Gallery opens displaying a list of mashups. This list is cached and refreshes periodically. Use **Refresh** to manually refresh this list.

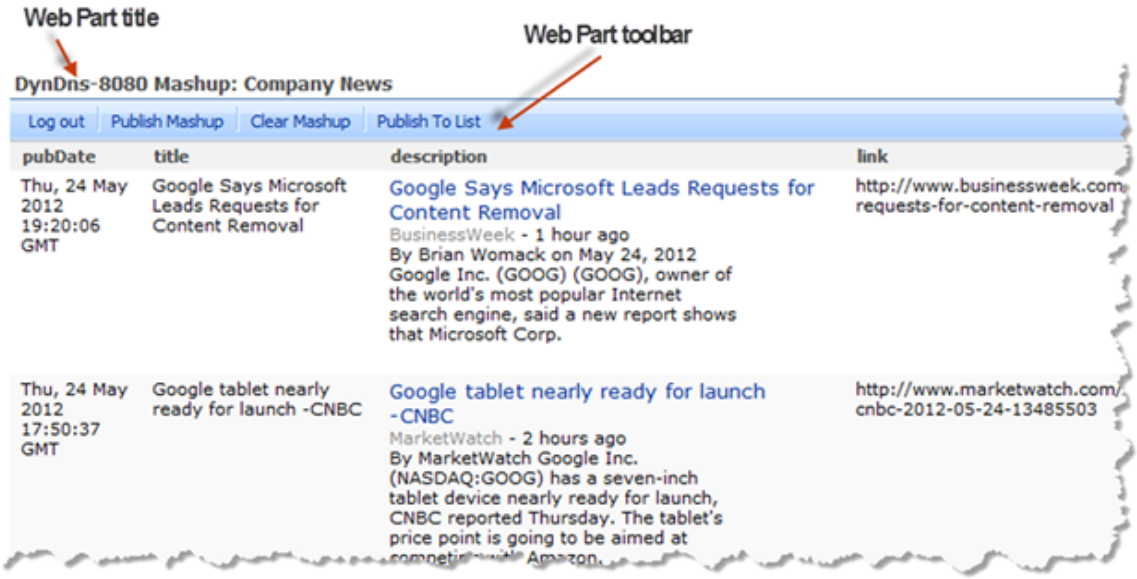


3. Find and select the mashup you want to add to this Web Part.

   You can filter the list of available mashups by category, by provider or enter a partial name in the search field.

4. If the mashup has input parameters, update these parameters as needed.

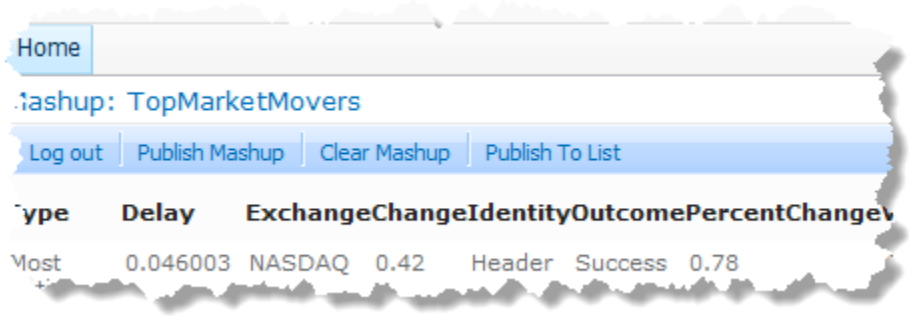
5. Click **Finish**.



The mashup you selected displays in the zone for this Web Part and, if you have appropriate permissions, includes a toolbar with actions for this mashup. See "Mashup Web Part Toolbar" on page 1423 for more information. See also "Update Mashup Web Part Properties" on page 1424 for other tasks you can perform, such as changing input parameters.

Mashup Web Part Toolbar

If you have permission to edit a Web Part Page, the mashup displays a toolbar with editing actions. The actions that appear in the toolbar change based on the status of the Web Part.



Actions on this toolbar include:

- **Log Out**: ends your session with the MashZone NextGen Server for this mashup. You can then log back in using different credentials. Simply click **Log Out**, enter another username and password and click **Log In**.

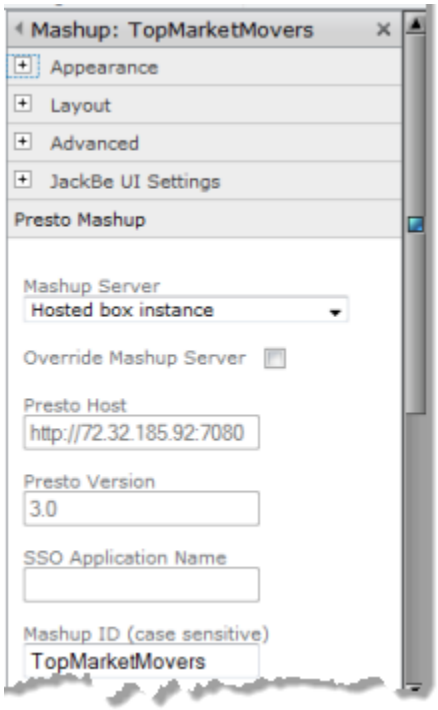
- **Publish Mashup**: allows you to find and add a different mashup to this Mashup Web Part. See "Publish a Mashup to a Mashup Web Part" on page 1421 for instructions.
- **Clear Mashup** removes the current mashup in this Mashup Web Part. Use **Publish Mashup** to find and add a different mashup.
- **Publish to List**: creates a SharePoint list in the current site with a snapshot of the data currently shown in this mashup. This creates a view named `MashZone NextGen` for this new SharePoint list. The title for the new list is the mashup's name. If a list with that title already exists in the current site, a Date/Time stamp is added to distinguish the two.

Update Mashup Web Part Properties

You can update Web Part properties to make changes to the appearance, layout and advanced features of the Web Part. Changes you make to the Web Part properties are visible to anyone viewing the published Web Part Page.

To edit Mashup Web Part properties, click the down arrow in the right corner of the Mashup Web Part title bar and select **Edit Web Part**.

This puts the page in Edit Mode and opens the Web Part Properties pane for this mashup:



In addition to the standard Web Part properties, Mashup Web Parts have two additional categories or properties: JackBe UI Settings andMashZone NextGenMashup. Use the properties in these categories to:

- "Change the Web Part Title" on page 1425
- "Change the Mashup Server Connection for a Mashup Web Part" on page 1426
- "Change the Mashup ID " on page 1426

See also "Update Mashup or App Input Parameters" on page 1434.

Change the Web Part Title

You control whether the Web Part title is automatically generated or is editable with the **Override Web Part Title** option in the JackBe UI Settings category of the Web Part properties.

This option is set by default which causes the Web Part title to be generated automatically in the form *mashup-server* `Mashup:` *mashup-title* .

1. Clear this option to be able to enter your own title for the Web Part.
2. To change the title, once you have cleared this option, expand the **Appearance** category and enter a new title.
3. Then click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change the Mashup Server Connection for a Mashup Web Part

You can change which MashZone NextGen Server this Web Part connects to using these Web Part properties in theMashZone NextGenMashup category:

- **Mashup Server**: select another MashZone NextGen Server from the list of servers that are visible in the current site. When you select a MashZone NextGen Server, this also sets theMashZone NextGenURL,MashZone NextGenversion and SSO Application ID properties. This also determines whether SSO authentication (2010)/MOSS SSO (2007) or cookie forwarding are applicable.

  Note: The single sign on feature in MashZone NextGen Add-On for SharePoint works with the SharePoint Secure Service Store service (2010) or MOSS SSO (2007) to enable users to automatically log in to multiple MashZone NextGen Servers without having to sign in to each.

- **Override Mashup Server**: this option is not set by default. Set this option to override the current MashZone NextGen Server connection information.
- **MashZone NextGen Host**: this property is only accessible if you have set the override property. Change this to the URL for the MashZone NextGen Server this Web Part should connect to. Enter the URL in the form:

  `http://`*mashup-server-address-or-domain* `:`*port*

  Note: You cannot use the localhost domain name in MashZone NextGen Server URLs.

- **MashZone NextGen Version**: this property is only accessible if you have set the override property. Enter the MashZone NextGen version number in use in the MashZone NextGen Server that this Web Part should connect to.
- **SSO Application Name**: this property is only accessible if you have set the override property. Enter an SSO application name here to enable SSO support when connecting to this MashZone NextGen Server.

  SSO applications (2010)/MOSS SSO applications (2007) are configured by the SharePoint farm administrator. If SSO is not enabled, you and other users of the published page must enter MashZone NextGen credentials when viewing the Mashup Web Part.

Then click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change the Mashup ID

This task should only be performed by developers or advanced users. You can change which mashup displays in the Web Part by changing the **Mashup ID** property in theMashZone NextGenMashup category.

Note: Mashup IDs are not identical with mashup names.

Enter the MashZone NextGen ID of the mashup to display in this Web Part. This ID must exist in the MashZone NextGen Server defined in Web Part properties.

Then click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

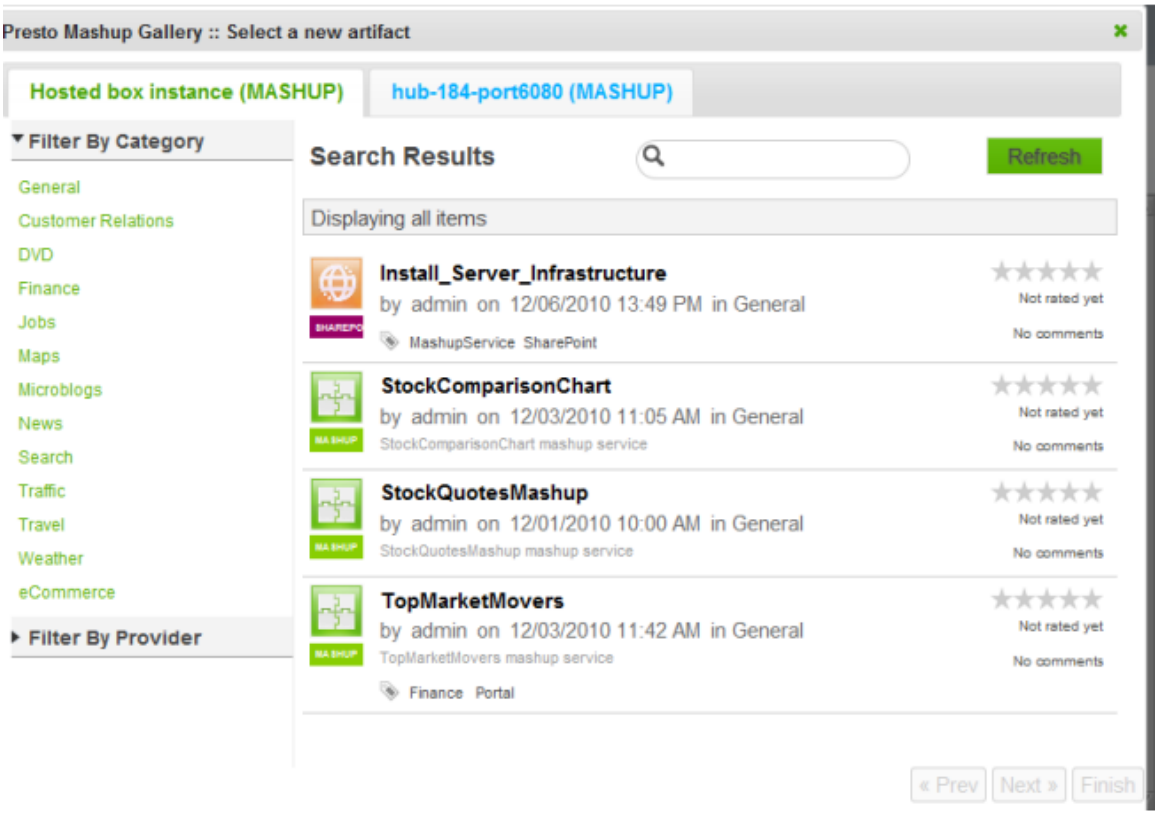
Publish Mashup Results as Native SharePoint Lists

You can create a copy mashup data as a native SharePoint list without using either Web Parts or External Content Types. This is best for mashups where results rarely or never change as the information is not updated in SharePoint once the list is created. The list does, however, fully participate in search or other integration techniques in SharePoint.

To publish a mashup directly as a native SharePoint list

1. In Site Actions, choose **More Options** (2010) or **Create** (2007).
2. In the Create dialog, choose **Lists >MashZone NextGenMashup List** and click **Create**.
3. Click **Select Mashup**, find the mashup you want in the MashZone NextGen Mashup Gallery and select it.

   The Mashup Gallery opens displaying a list of mashups. This list is cached and refreshes periodically. Use **Refresh** to manually refresh this list.



4. If the mashup has input parameters, update these parameters as needed. Click **Finish**.

   Note: Review the input parameters values carefully as you cannot change this once you have selected the mashup.

5. Complete these properties for the List:

   - **List Title** defaults to the name of the mashup. Update this if needed.
   - Enter a **Description** for the List.
   - Choose one of the mashup fields as the **Title Field** if you want that column to be the title for each item in the List.
   - If prompted, enter the appropriate credentials to log into the MashZone NextGen Server for this mashup.

6. Click **Publish**.

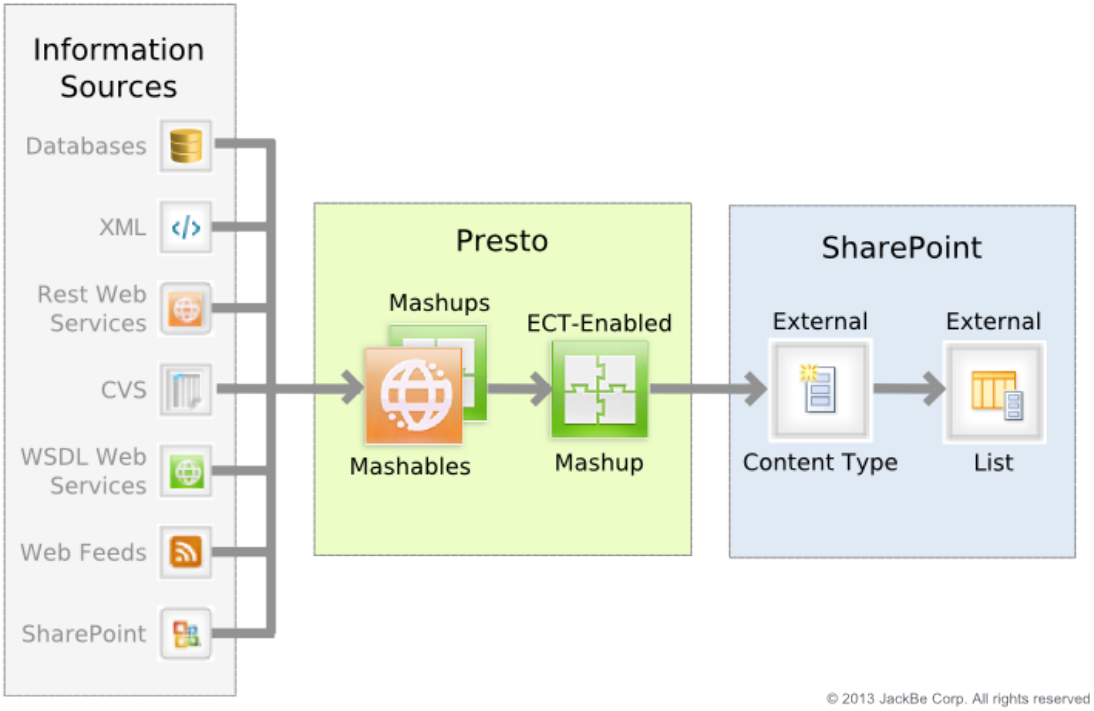
   The new list opens displaying each item from the mashup. This list is a copy of the mashup result when it is created. The list is not updated if mashup results change, nor can you update input parameters for the mashup, if any.

Publish Mashup Results as External SharePoint Lists

You can integrate information from many different external systems in your SharePoint 2010 sites using MashZone NextGen mashups as *external content lists*. If information of interest comes from a MashZone NextGen mashable, simply wrap the results in a compatible mashup in MashZone NextGen.

Important: This feature is *not* available for SharePoint 2007.

MashZone NextGen makes it simple to connect to many different types of information sources in your organization including databases, web services (using WSDLs or REST), web feeds (using RSS or Atom), CSV or XML. You can expose this data and integrate it in your SharePoint sites using MashZone NextGen mashups and SharePoint external content lists.



Consider the Requirements and Limitations and then follow The Mashup-to-External-List Process to get started.

Requirements and Limitations

You use MashZone NextGen Wires to create a mashup that SharePoint can access as an external content type. You use SharePoint Designer to define the external content type for mashups that are *ECT-enabled* and the external content lists based on these external content types.

Both MashZone NextGen and SharePoint must be configured properly to support communication between SharePoint and MashZone NextGen and between SharePoint Designer and MashZone NextGen. MashZone NextGen and SharePoint administrators must:

- *Enable one-way SSL* for clients connecting to the MashZone NextGenMashZone NextGen Server. This is required to allow SharePoint Designer to use a secure connection when you create external content types and external content lists based on MashZone NextGen mashups.

  MashZone NextGen administrators should see One-Way SSL to MashZone NextGen for instructions.

- *Use the Secure Service Store* in SharePoint to store credentials for MashZone NextGen used in connections from SharePoint and SharePoint Designer to these mashups.

  SharePoint administrators should see Configure Authentication for P4S 2010 for instructions on setting up SSO + Token Authentication.

In addition to basic connection requirements, there are specific requirements and limitations on the mashup results that you can use in external content lists and on the maintenance that is possible to these mashups. See "Supported SharePoint Operations" on page 315, "Data Requirements for the Results of ECT-Enabled Mashups" on page 316 and "Updates to the Mashup's Results" on page 317 for more information.

The Mashup-to-External-List Process

The steps to connect to an external system and integrate this dynamic information as using MashZone NextGen mashups are:

1. In MashZone NextGen Hub, create a mashable or mashup to connect to the external system with the information you want to use in a SharePoint list. See "Connect Information Sources as Mashables" on page 136 and "Creating Mashups in Wires" on page 284 for links and more information.
2. In Wires (in MashZone NextGen Hub), create a mashup that adapts the mashable or mashup you created in the previous step to allow SharePoint to use this mashup as an external content type. See "Adapt Results as a SharePoint External Content Type" on page 314 for instructions.
3. In SharePoint Designer, Create an External Content Type for a Mashup using the mashup created in the previous step.
4. In SharePoint Designer, Create an External Content List from a Mashup using the external content type created in the previous step to connect to the ECT-enabled mashup in MashZone NextGen.

   Save this list, add views or use any feature in SharePoint or SharePoint Designer to integrate this external list in your site.

Create an External Content Type for a Mashup

External content types define how to connect to a data source that is external to SharePoint and retrieve or perform other operations on specific information. It also defines metadata about the information itself, such as column name and datatypes.

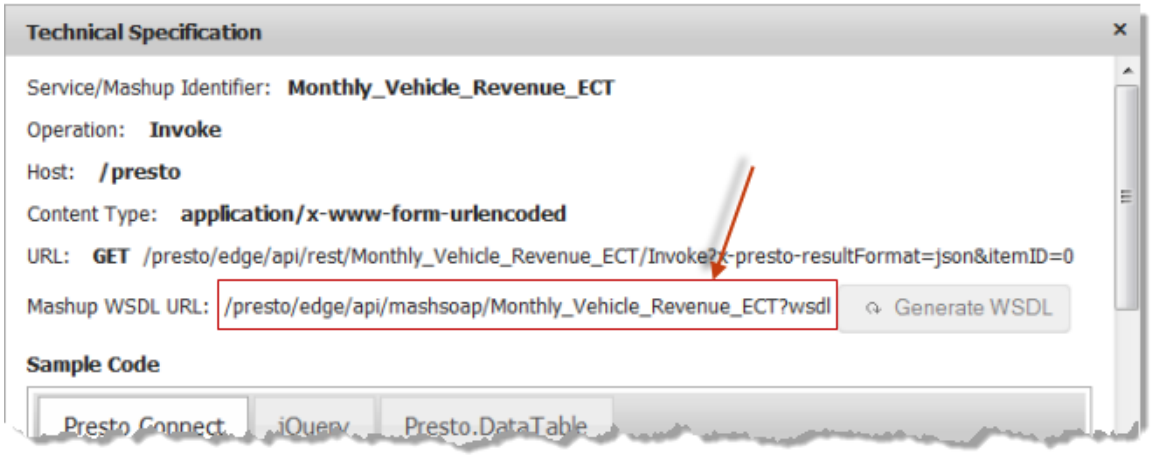
You use SharePoint Designer to customize your SharePoint sites, including defining external content types and creating external content lists. SharePoint Designer is a free tool that you download and install separate from SharePoint.

The basics of defining an external content type in SharePoint Designer are discussed in online help for SharePoint Designer, which covers how to create an external content type for a database. The process to create an external content type for mashups is slightly different:

1. Before you begin working in SharePoint Designer, open the ECT-enabled mashup that you want to work with in MashZone NextGen Hub. You must be the mashup owner or a MashZone NextGen administrator or developer to complete this step.

   a. Click **Show >** **Technical Spec** from the mashup's artifact page.

   b. Find the Mashup WSDL URL and copy it. You will use this as you create an external content type for this mashup.



2. Open SharePoint Designer and open the site where you want to use mashup results.

   a. Select external content types and add a new content type.

   b. Enter a name for this external content type. Click the link for **External System**.

   c. Click **Add Connection**.

3. Choose `WCF Service` as the data source type.

4. Use the mashup WSDL URL you copied in step 1 as the **Service Metadata URL** and *change* the parameter at the end of the url to `sp_wsdl`.

   If the URL from the mashup technical spec is:

```
https://myOrg.mservers.com:8443/presto/edge/api/rest/
Monthly_Vehicle_Revenue_ECT?wsdl
```

Then change this to:

```
https://myOrg.mservers.com:8443/presto/edge/api/rest/
Monthly_Vehicle_Revenue_ECT?sp_wsdl
```

5. Use the mashup WSDL URL you copied in step 1 as the **Service Endpoint URL** and *remove* the `?wsdl` parameter from the URL. With the WSDL URL shown in the previous step, the service endpoint would be:

   ```
   https://myOrg.mservers.com:8443/presto/edge/api/rest/
   Monthly_Vehicle_Revenue_ECT
   ```

6. Change the WCF Service Authentication Settings to **Connect with Impersonated Custom Identity**.

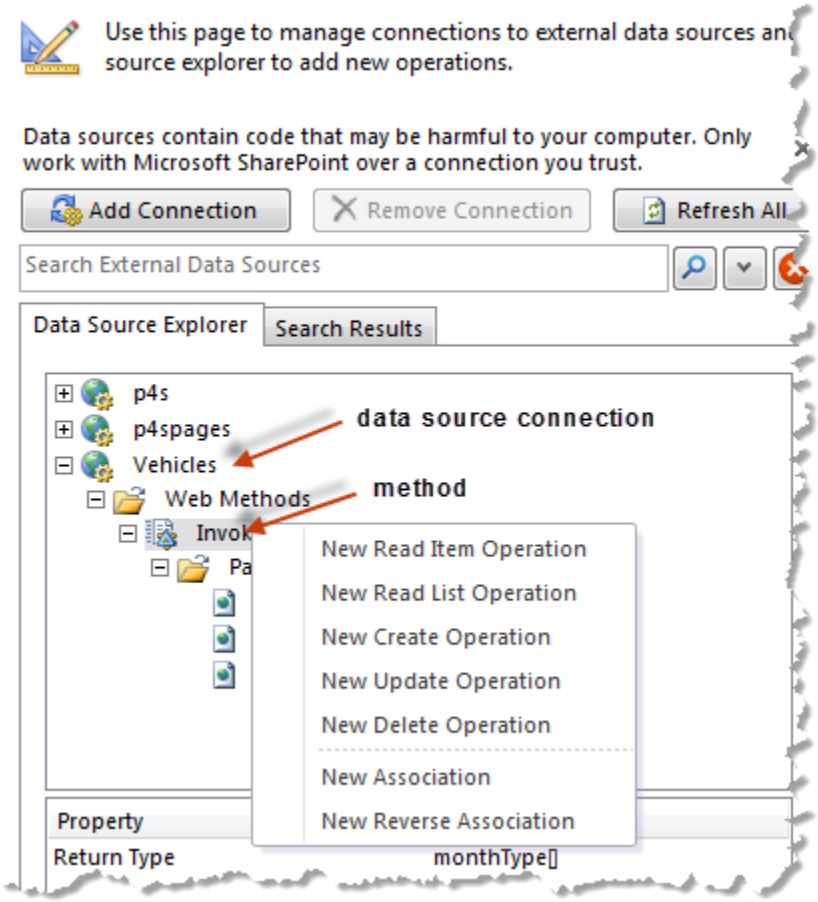
7. Enter the name of the **Secure Service Application ID** for the MashZone NextGen Server that hosts this mashup.

   Contact your SharePoint administrator to get the correct application ID for this field.

8. Update any other settings and click **OK**.

   Note: In some cases, you may get an error loading URLs as SharePoint validates both the WSDL and endpoint URLs. See "Troubleshooting URL Loading Problems" on page 1433 for suggestions.

9. The connection you created should now be listed. Expand this till you see the `Invoke` method.



10. Right click the method and create a **New Read Item Operation**.

    This opens a wizard to define this operation. You can change names or display names and other properties as needed. You *must* map both the input parameter and the return parameter used as the item ID to the Identifier for lists of this type.

11. Right click the method again and create a **New Read List Operation**.

12. Once these two operations are defined, save this external content type.

You can now add external content lists based on this type.

Create an External Content List from a Mashup

Once you have an external content type defined for an ECT-enabled mashup, you can add external content lists with mashup results to SharePoint sites:

1. In SharePoint Designer, open the site you want to work in and select External Content Types.

2. Select the external content type for the mashup you want to use and click **Create Lists and Forms** in the ribbon.

3. Give the list a name, choose the operation to get an individual list item and save this list.

   No forms are needed unless the mashup supports create, update or delete operations.

Troubleshooting URL Loading Problems

When you create new connections, you may receive errors trying to load the WSDL URL or the endpoint URL. The most common reasons for these errors are:

- Minor typos in the URLs.
  - Make sure the URL for the WSDL ends in `?sp_wsdl`.
  - Make sure the endpoint URL has *no* parameters (`?sp_wsdl`) at the end.
- The wrong Secure Service Application ID.

  This application stores MashZone NextGen credentials that are mapped to different SharePoint users. Check with your SharePoint administrator and verify that:
  - The application ID is correct.
  - Your MashZone NextGen credentials exist in this application and are correct.
  - This MashZone NextGen account has permission to use this mashup.
- A missing certificate for this MashZone NextGen Server.

  SharePoint Designer uses a secure connection to the MashZone NextGen Server. You must install the digital certificate for this MashZone NextGen Server on your computer to connect successfully.

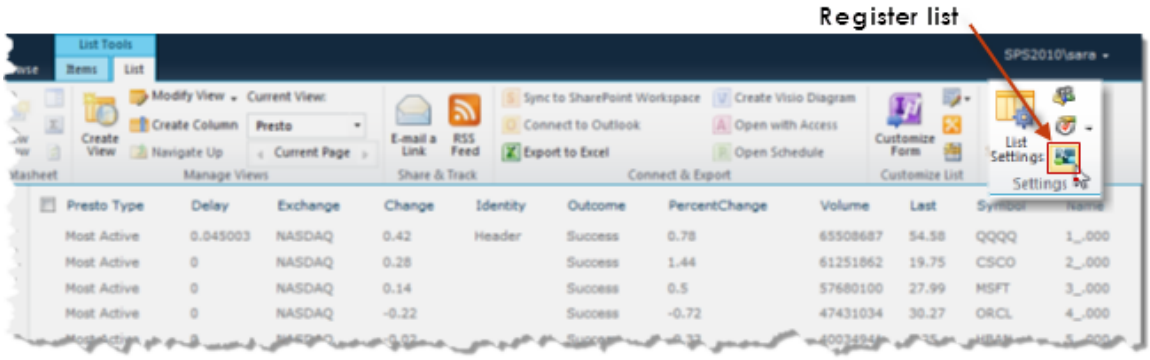
  1. Contact your MashZone NextGen administrator to obtain the correct certificate. Typically, this is a public.cer file.
  2. Double click this file to start the wizard to install this certificate.
  3. Do *not* install the certificate in the default or personal certificate store. Set the option to choose the certificate store.
  4. Select `Trusted Root Certificate Authorities` as the certificate store.

Register a SharePoint List as a MashZone NextGen Mashable

You can use information from SharePoint Lists in MashZone NextGen mashups or apps by registering Lists as a *mashable information source*.

To register a List in MashZone NextGen

1. In SharePoint 2010, open a list and click **List** in the List Tools tab.
2. In SharePoint 2007, open a list and click *Actions*.
3. Select **Register List** in the ribbon toolbar.



4. If desired, change the **Name** or **Description** to assign to theMashZone NextGenmashable. The name defaults to the List title.
5. If this List has views, choose the specific view you want to use for this MashZone NextGen mashable.
6. In **Publish To**, select the MashZone NextGen Server where you want to register this List.
7. If desired, change **SharePoint Access Credentials** to control which user credentials are used in MashZone NextGen to retrieve information from this List:
   - **Bind now** = set this option to always use the credentials you have supplied while registering this SharePoint list as a mashable.
   - **Bind at runtime dynamically** = this is set by default. With this option, MashZone NextGen uses the credentials of the current user to access information for this SharePoint List, rather than your credentials.
8. Click *Submit* to reigster the List as a mashable.

Update Mashup or App Input Parameters

Input parameters allow mashups and apps to produce and display information that is tailored to your specific needs. For example, a mashup to find businesses within a given location might have input parameters for the type of business to find and for the city and state to use as the location. Input parameters may also determine how to filter information or how to sort it, among many other functions.

Not all mashups or apps have input parameters, not all input parameters are required. You can update input parameters for mashups or apps in:

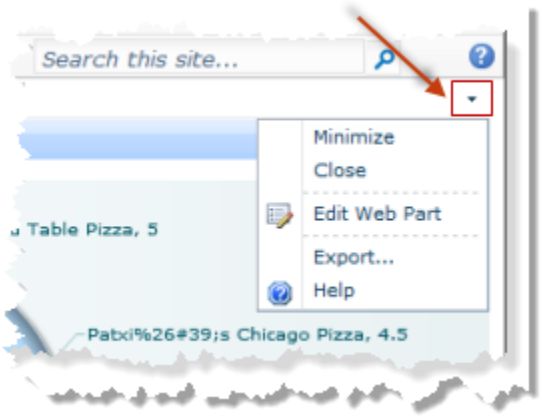
- The Mashup Gallery or App Gallery when you publish mashups or apps to Web Parts.
- The app itself, from a toolbar button in the app title bar.

  This title bar is hidden, by default, but you can update the Web Part to show this instead. See "Hide or Show App or View Titles and Toolbars" on page 1443 for more information. If this title bar is visible, any user who has permissions to use the app can update input parameters.

- The Web Part properties pane for the Mashup or App Web Part.

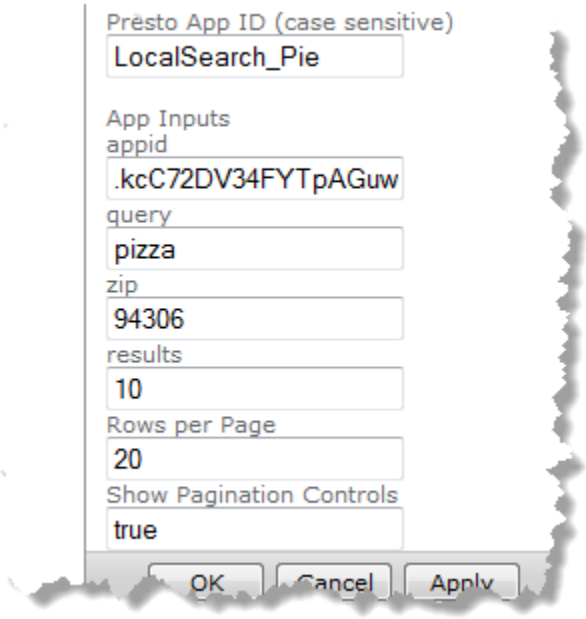
To edit input parameters from Web Part properties

1. Click the down arrow in the right corner of the Mashup or App Web Part title bar and select **Edit Web Part**.



   This puts the page in Edit Mode and opens the Web Part Properties pane for this mashup or app.

2. Find the input parameter fields in the **MashZone NextGen** or **MashZone NextGen** category.



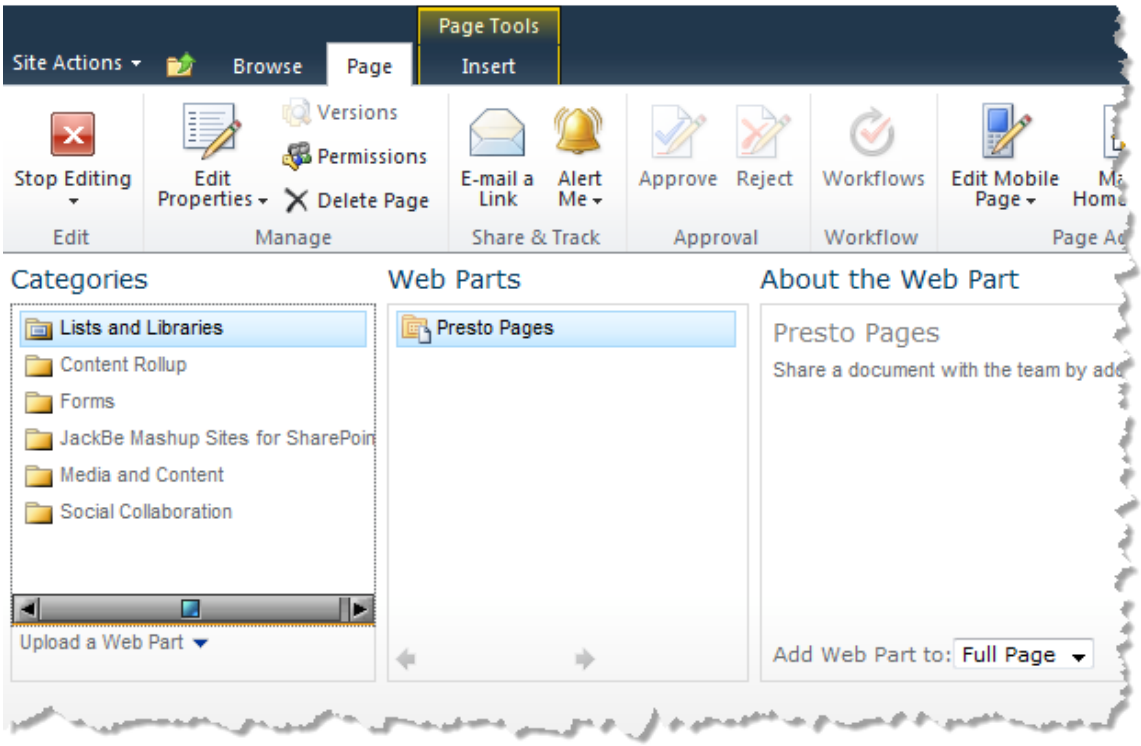
3. Update the input parameter values as needed.
4. For some apps, you can also Change Pagination or Themes.

5. Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

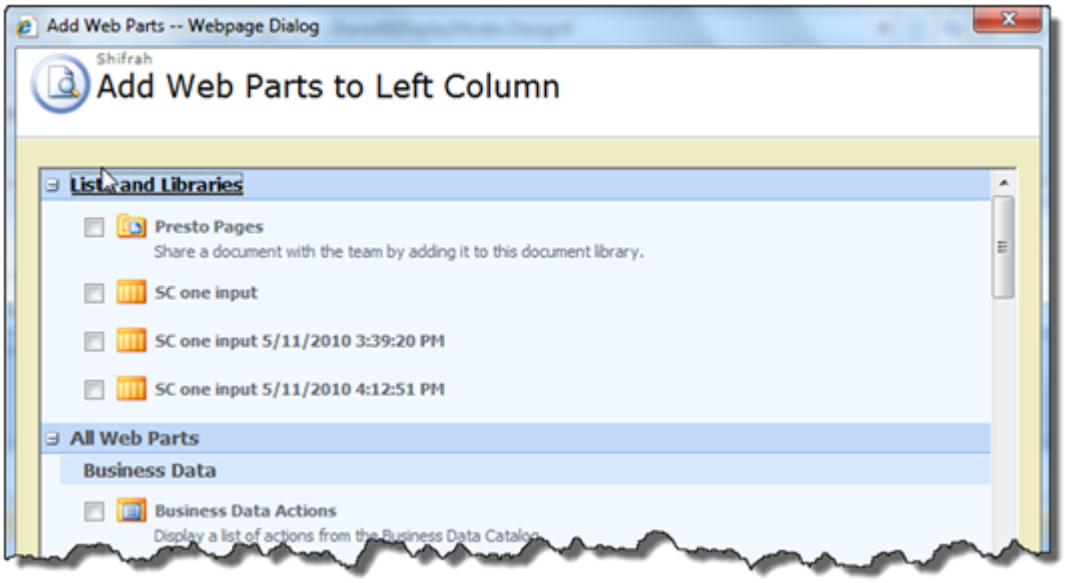
Add an App Web Part to the Web Part Page

In order to publish apps to a Web Part Page, you must add one or more app Web Parts to the zones of that page. You then publish apps to each App Web Part.

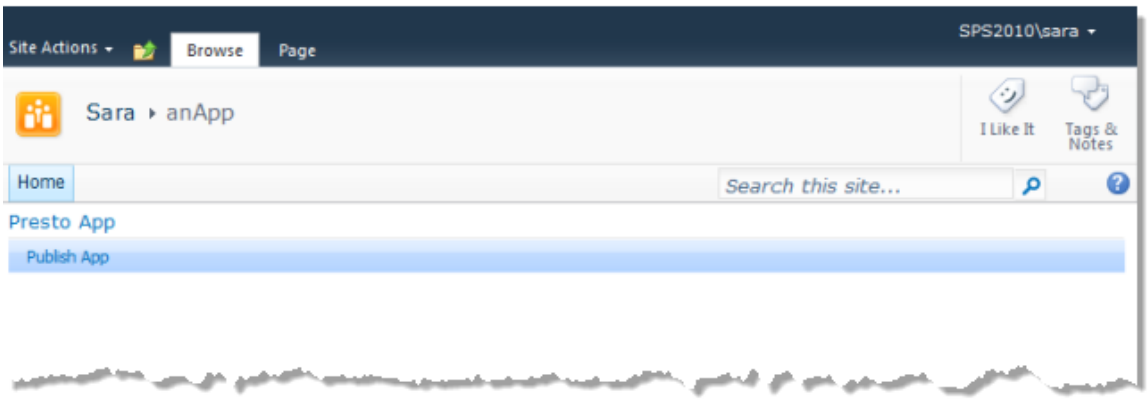
1. In SharePoint 2010, click **Add a Web Part** in the zone on your Web Part page where you would like to place an app. The page opens in edit mode with a list of categories to choose from:



2. In SharePoint 2007, Click **Add a Web Part** in the zone on your Web Part page where you would like to place an app. The Add Web Part menu opens.



3. Select the **JackBe Mashup Sites for SharePoint** category and set the **App Web Part** option.
4. Click **Add** (in the bottom right corner).
5. Continue editing this Web Parts page. You can:
   - Add appropriate Web Parts to other zones on the page.
   - Edit the properties for your App Web Part, or any Web Part on the page. See "Update App Web Part Properties" on page 1441 for more information.
   - Select and drag Web Parts to move them to other zones.
6. Once you are finished adding and modifying Web Parts, click **Stop Editing** in the Web Part page toolbar. The page returns to the normal display mode, with an App Web Part and toolbar in the zones you have selected.



You can now Publish an App to an App Web Part.

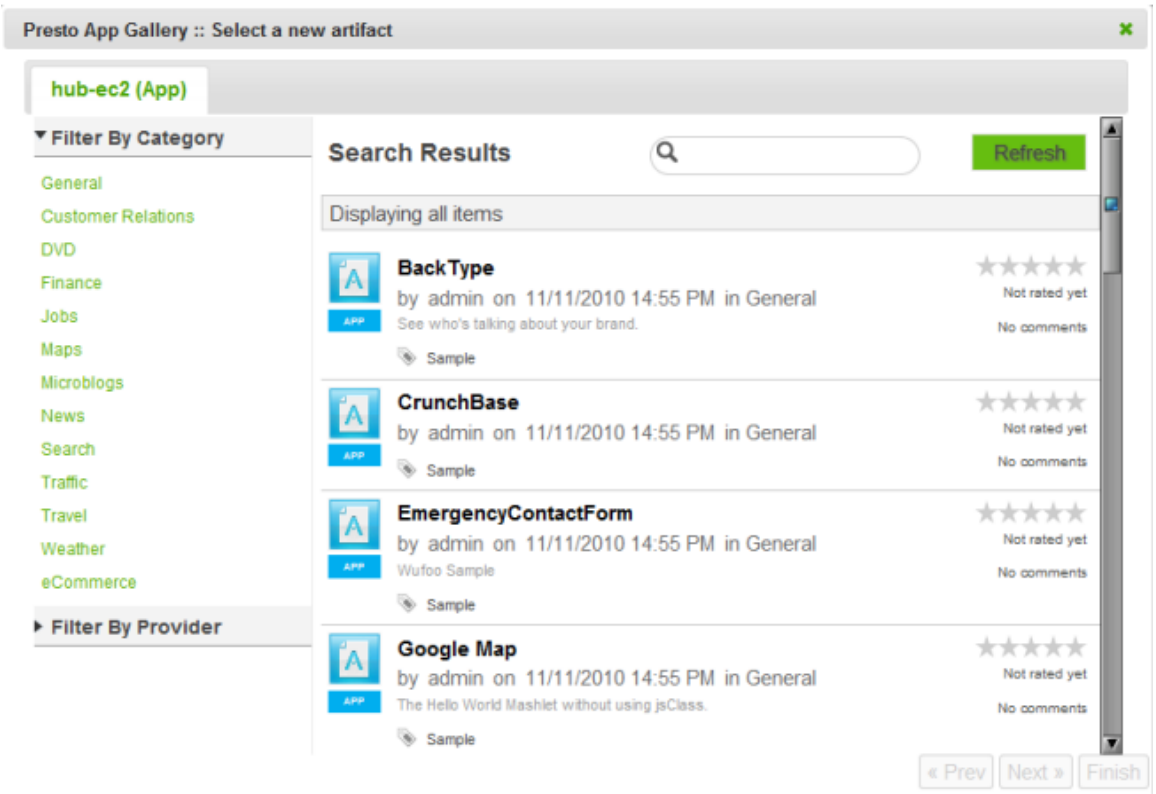
Publish an App to an App Web Part

Once you have stopped editing your Web Part page, you can select the specific app to use in any App Web Parts on your page.

1. Click **Publish App** from the toolbar in any of the App Web Parts on your page.
2. If prompted, enter credentials to log in to the MashZone NextGen Servers you want to work with. You can also **Skip** specific MashZone NextGen Servers. Apps from the MashZone NextGen Servers that you skip will not be listed.

   Note: Do *not* include the Windows domain in your username, such as `mydomain\myname`, when you log in. The MashZone NextGen Server does not accept domain names. You may also need to add MashZone NextGen Servers to your trusted sites or change your cookie preferences to allow session cookies. If you still receive login errors, check with your MashZone NextGen administrator.

   The App Gallery opens displaying a list of apps. This list is cached and refreshes periodically. Use **Refresh** to manually refresh this list.

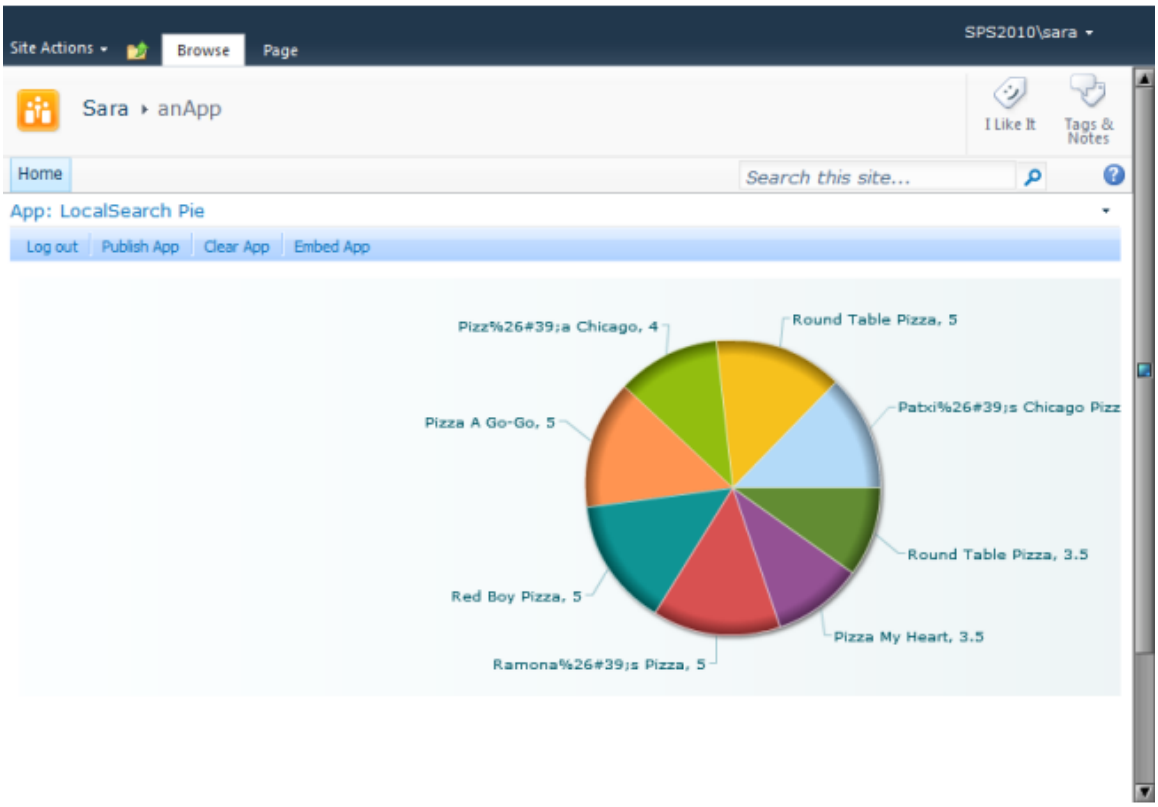


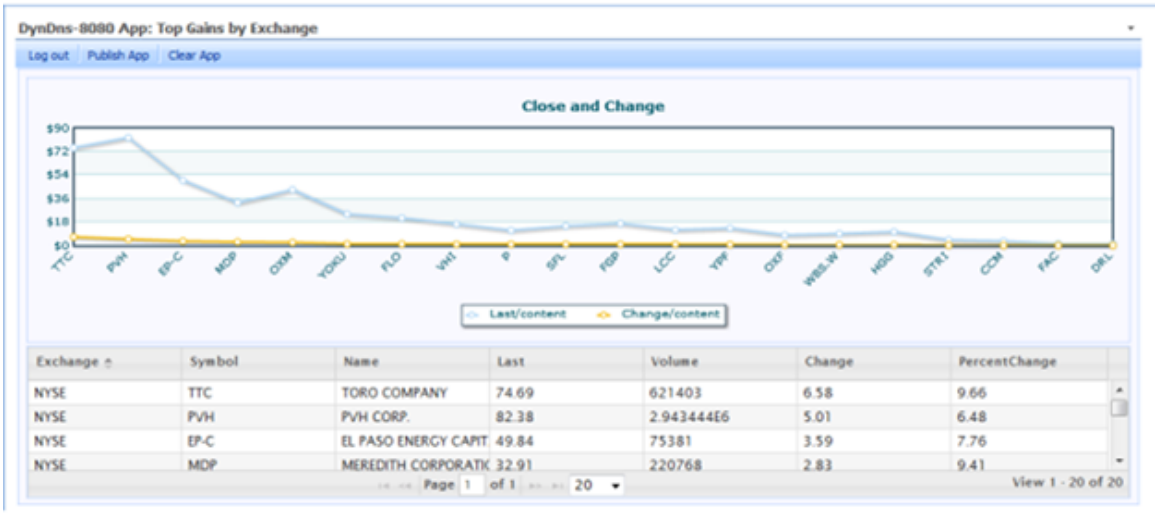
3. Find and select the app you want to add to this Web Part.

You can filter the list of available apps by category, by provider or enter a partial name in the search field.

4. If the app has input parameters, update these parameters as needed.
5. Click **Finish**.

   The app you selected displays in the zone for this Web Part.

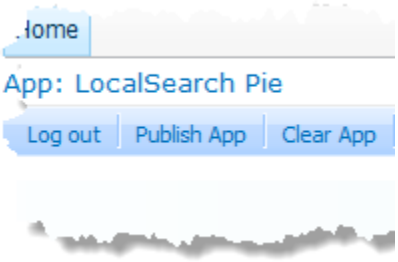




If you have appropriate permissions, this includes a toolbar with actions for this app. See "App Web Part Toolbar" on page 1441 for more information. See also "Update App Web Part Properties" on page 1441 for other tasks you can perform.

App Web Part Toolbar

If you have permission to edit a Web Part Page, the app displays a toolbar with editing actions. The actions that appear in the toolbar change based on the status of the Web Part.



Actions on this toolbar include:

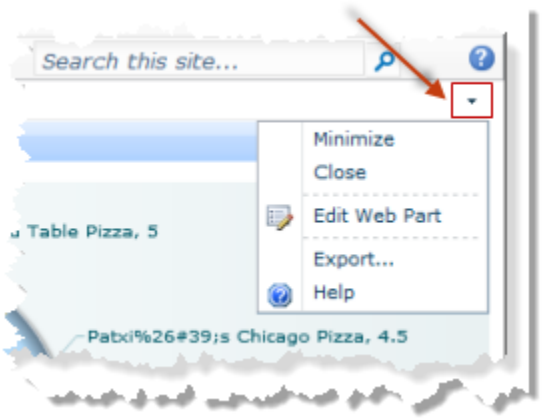
- **Log Out**: ends your session with the MashZone NextGen Server for this app. You can then log back in using different credentials. Simply click **Log Out**, enter another username and password and click **Log In**.
- **Publish App**: allows you to find and add a different app to this App Web Part. See "Publish an App to an App Web Part" on page 1438 for instructions.
- **Clear App** removes the current app in this App Web Part. Use **Publish App** to find and add a different mashup.

In SharePoint 2007, you may hide or show the App Web Part Toolbar with **Site Actions >Hide Web Part Menu Bar** or **Show Web Part Menu Bar**.

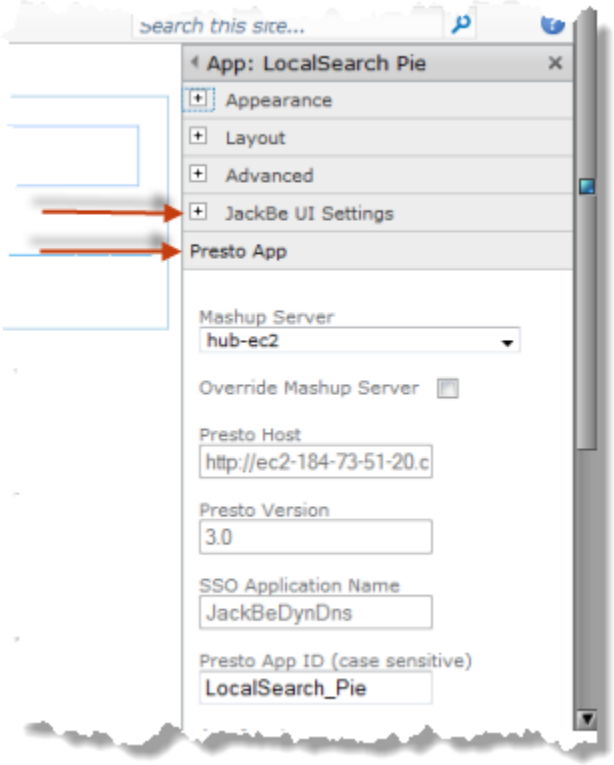
Update App Web Part Properties

You can update Web Part properties to make changes to the appearance, layout and advanced features of an App Web Part. Changes you make to the Web Part properties are visible to anyone viewing the published Web Part Page.

To edit App Web Part properties, click the down arrow in the right corner of the App Web Part title bar and select Edit Web Part.



This puts the page in Edit Mode and opens the Web Part Properties pane for this app:



In addition to the standard Web Part properties, app Web Parts have two additional categories of properties that you can update: JackBe UI Settings and MashZone NextGen App. Use these properties to:

- Change the Web Part Title
- Hide or Show App or View Titles and Toolbars
- Change Pagination or Themes
- Enable Anonymous Access

- Change the Mashup Server Connection for an App Web Part
- Change the App ID

See also "Update Mashup or App Input Parameters" on page 1434.

Change the Web Part Title

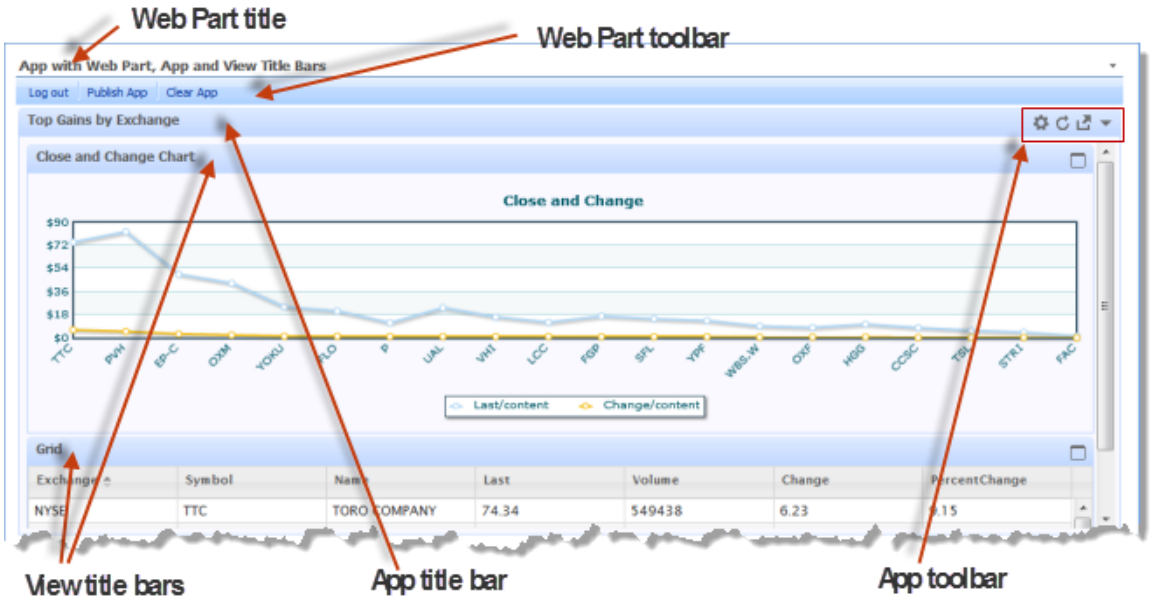
You control whether the Web Part title is automatically generated or is editable with the **Override Web Part Title** option in the JackBe UI Settings category of the Web Part properties.

This option is set by default which causes the Web Part title to be generated automatically in the form *mashup-server* `App:` *app-title* . To change this default title:

1. Clear the **Override Web Part Title** option in the JackBe UI Settings category.
2. Expand the **Appearance** category and enter a new title.
3. Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Hide or Show App or View Titles and Toolbars

Both the App Title Bar and the View Title Bars contain titles and toolbars with buttons to assist users, as shown in this example:



These title bars do take additional space, but also can provide useful functionality. The toolbar in the App Title Bar allows users to update input parameters at any time for apps that have them. It also allows uers to refresh the app at any time. The toolbar in View Titles Bars allows users to maximize a specific view to see it in more detail or show all views (the default).

By default both the App Title Bar and the View Title Bars are hidden when you publish an app in Web Part. You can show or hide each of these title bars using these two Web Part properties in the JackBe UI category:

- **Disable Header**: hides or shows the App Title and toolbar. Clear this option to show this title and toolbar.
- **Disable View Header**: hides or shows the View Titles and toolbars for apps that have multiple view. Clear this option to show these titles and toolbars.

Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change Pagination or Themes

Some apps have the following properties that determine how much data is shown in the app and the color theme used for borders and title bars in the app:

- **Enable pagination**: For apps with lots of data. showing smaller subsets of data provides better performance and can make it easier to find specific data of interest. If this property is set, the app displays just one 'page' of data. It also includes a toolbar to allow users to move to other pages of data.
- **Rows Per Page**: This only applies if pagination is enabled. This determines how many rows of data are included in one page.
- **Theme**: Defines a color scheme for borders for the app and, if shown, for the App and View Title bars. See "Hide or Show App or View Titles and Toolbars" on page 1443 for more information.

Enable Anonymous Access

Anonymous access allows guest users who are not logged in to view and work with an app. You only need to set the *Enable Anonymous Access* option for an App Web Part if guest users should have access and neither SSO nor cookie forwarding is in use with this site to automatically provide user credentials.

Change the Mashup Server Connection for an App Web Part

You can change which MashZone NextGen Server this Web Part connects to using these Web Part properties in the MashZone NextGen App category:

- **Mashup Server**: select another MashZone NextGen Server from the list of servers that are visible in the current site. When you select a MashZone NextGen Server, this also sets the MashZone NextGen URL, MashZone NextGen version and SSO Application ID properties. This also determines whether SSO authentication and cookie forwarding are applicable.

  Note: The single sign on feature in MashZone NextGen Add-On for SharePoint works with the SharePoint Secure Store service to enable users to

automatically log in to multiple MashZone NextGen Servers without having to sign in to each.

- **Override Mashup Server**: this option is not set by default. Set this option to override the current MashZone NextGen Server connection information.
- **MashZone NextGen Host**: this property is only accessible if you have set the override property. Change this to the URL for the MashZone NextGen Server this Web Part should connect to. Enter the URL in the form:

  `http://`*mashup-server-address-or-domain* `:`*port*

  Note: You cannot use the localhost domain name in MashZone NextGen Server URLs.

- **MashZone NextGen Version**: this property is only accessible if you have set the override property. Enter the MashZone NextGen version number in use in the MashZone NextGen Server that this Web Part should connect to.
- **SSO Application Name**: this property is only accessible if you have set the override property. Enter an SSO application name here to enable SSO support when connecting to this MashZone NextGen Server.

  SSO applications are configured by the SharePoint farm administrator. If SSO is not enabled, you and other users of the published page must enter MashZone NextGen credentials when viewing the Mashup Web Part.

Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

Change the App ID

This task should only be performed by developers or advanced users. You can change which app displays in the Web Part by changing the **App ID** property in the MashZone NextGen App category.

Note: App IDs are not identical to app names.

Enter the MashZone NextGen ID of the app to display in this Web Part. This ID must exist in the MashZone NextGen Server defined in Web Part properties.

Click **OK** to save your changes and close the Web Part properties task pane or click **Apply** to save your changes and continue updating properties.

13 MashZone NextGen Development and APIs

- Features of the MashZone NextGen REST or MashZone NextGen Connect APIs 1479
- Working with theMashZone NextGenREST API 1480
- Use MashZone NextGen Connect for Javascript 1484
- Use Mashable/Mashup Technical Specs 1489
- Paginate Mashable or Mashup Responses 1491
- Map MashZone NextGen Attributes to Artifact Input Parameters 1502
- MashZone NextGen APIs 1507
- MashZone NextGen Platform API Console 1529
- MashZone NextGen APIs 1536
- MashZone NextGen Platform API Console 1558

MashZone NextGen developers create artifacts and extend MashZone NextGen capabilities to support the business requirements of their organization and enable power users. In addition to creating mashups or apps in visual tools such as Wires, developers write code and work in technical editors such as the Mashup Editor.

These developer-specific tasks can include:

- Creating mashups and macros using EMML. See "Mashups in EMML" on page 441 for links to more information on creating mashups or macros.
- Connecting to MashZone NextGen to invoke artifacts using the MashZone NextGen REST API or the MashZone NextGen Connect API. Typically, you invoke mashables, mashups or use snapshots in custom apps, but this may also include other types of clients.

  Note: To help determine which API fits your needs, see "Features of the MashZone NextGen REST or MashZone NextGen Connect APIs" on page 1479 for a comparison.

  To connect to the MashZone NextGen Server and invoke artifacts, you use one of these APIs:

  - The "MashZone NextGen REST API" on page 1509. See "Working with theMashZone NextGenREST API" on page 1480 for an introduction and simple example.
  - The MashZone NextGen Connect for JavaScript (PC4JS). See "Use MashZone NextGen Connect for Javascript" on page 1484 for an introduction and simple example. See the MashZone NextGenConnect for JavaScript API for more information on the API.
  - The "MashZone NextGen Snapshot API" on page 1517. This is a REST API specifically for retrieving snapshots.

  See also "Paginate Mashable or Mashup Responses" on page 1491 and "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502.
- Creating pluggable views and plugging them into the MashZone NextGen View Gallery and View Maker (see "Creating Pluggable Views or Libraries" on page 1045 for more information and links) using the MashZone NextGenViews API and the MashZone NextGenDataTable API.
- Creating custom apps (see "Custom Apps" on page 1191 for links) using the App Editor, App Specifications and the MashZone NextGenApp API, MashZone NextGen Library Loader API and MashZone NextGenDataTable API.
- Extending functionality to simplify development, provide business-specific functionality to enable self-service for power users, or integrate MashZone NextGen within your environment. Extensions may include:
  - "Defining Custom XPath Functions" on page 601 for use in mashups or macros with the MashZone NextGenCustom XPath API

- "Adding Custom Blocks to MashZone NextGen Wires Using Macros" on page 392
- "Creating Pluggable Views or Libraries" on page 1045 using the MashZone NextGen Views API or using the "Template View" on page 1006 to provide more visual options for basic apps and workspace apps.
- Creating extensions to handle specific requirements for authentication with MashZone NextGen using the MashZone NextGenCustom Certificate Validation or MashZone NextGenCustom SSO Filter API.
- Creating custom security profiles (see "Implement a Custom Security Profile Client" on page 164) using the MashZone NextGenSecurity Profile API to enable secure connections with mashable information sources that have unique authentication requirements.
- Using the "MashZone NextGen Platform API Console" on page 1529 for other extensions or integration projects.

Features of the MashZone NextGen REST or MashZone NextGen Connect APIs

Features	REST API	PC4JS
Use the language and library or your choice	yes	
REST or JUMP protocol. The JUMP protocol from previous releases is supported in 3.0 and above, although REST is recommended as the preferred protocol for working with MashZone NextGen. For more information on JUMP, see the About JUMP topic.	REST	REST
Asynchronous or synchronous connections	language / library specific	asynch
Number of users/requests per connection	language / library specific	1 user / + requests
Supports named parameters and ordered parameters	named only	named only

Features	REST API	PC4JS
HTTP and HTTPS	language / library specific	yes
Can set HTTP headers in requests	language / library specific	
Can access local and remote MashZone NextGen Servers	yes	yes
Handles cross-domain requests (specific to remote servers and JavaScript)	requires additional code	yes
Handles authentication and session management	requires additional code	yes
Supports guest users (anonymous connections)	requires additional code	yes
Response formats	Native, JSON or XML, requires additional code	JSON or XML

Working with theMashZone NextGenREST API

You may use the MashZone NextGen REST API to invoke any mashable information source or mashup or retrieve snapshots to work with snapshot data. Use any language or library that supports HTTP or HTTPS. For detailed information, see the "MashZone NextGen REST API" on page 1509 reference.

To invoke a mashable or mashup:

1. "Get the Mashable/Mashup Specification" on page 1481
2. "Login or Set Guess Access" on page 1481
3. "Build and Send the Request" on page 1482

4. Handle security and session issues including: "Cross Domain Requests" on page 1483 and "Session Management" on page 1484

To work with snapshot data, see the "MashZone NextGen Snapshot API" on page 1517.

Get the Mashable/Mashup Specification

First, find the URL and other information needed to invoke the mashable or mashup. You can find this information in the Technical Spec tab on the artifact page for that mashable or mashup. For example:

See "Use Mashable/Mashup Technical Specs" on page 1489 for instructions and examples of mashables using HTTP GET or POST.

Login or Set Guess Access

Authentication requirements depends on how authentication is configured in MashZone NextGen:

- *With Single Sign-On*: authentication is handled automatically by the SSO solution. Login is not required and no additional information is needed in requests.
- *With Digital Certificates and SSL*: authentication is handled automatically with SSL. Login is not required and no additional information is needed in requests.
- *Anonymous Requests for Guest Access*: supply no user credentials, so that MashZone NextGen treats request as coming from a guest user. To send anonymous requests,

include the `x-p-anonymous` MashZone NextGen header/parameter in the URL of each request or as an HTTP header.

- *With Basic Credentials*: this is the default authentication mechanism when you install MashZone NextGen. In this case, you must use the REST API method to login to create a MashZone NextGen session and authenticate the user. You can also use the REST API method to logout.
  - For *Login* use this URL:

    `http://`*app-server*`:`*port*`/presto/edge/api/rest/UserManagerService/login?presto_username=`*username*`&presto_password=`*pw*

    Both parameters must be in clear text.
  - For *Logout* use this URL:

    `http://`*app-server*`:`*port*`/presto/edge/api/rest/UserManagerService/logout`

  Using jQuery, for example, you would login something like this:

```
jQuery(function() {
  var requestBody = '';
  jQuery.ajax({
    type: "get",
    contentType: "application/x-www-form-urlencoded",
    url: "/presto/edge/api/rest/UserManagerService/login?x-presto-resultFormat=json&presto_use
    userA&presto_password=pwA",
    data: requestBody,
    dataType: "text",
    success: function(response){
        var result = response;
        if(typeof response !== "string") {
           result = Object.toJSON(response);
        }
        //handle success
      },
    error: function(xhr, errorStatus, e) {
        //handle error
      }
  });
});
```

Build and Send the Request

Take the URL from the Technical Specs for the mashable or mashup and:

- Update any of the parameters, as needed.

  You can set parameter values programmatically, or use MashZone NextGen attributes to have the MashZone NextGen Server resolve parameter values. See "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502 for more information.
- Remove or add "MashZone NextGen Headers/Parameters" on page 1511 to control aspects of request processing or specific information in the response.

Some of the most common headers include:

- `x-p-resultFormat` to have MashZone NextGen send the result in CSV, JSON or XML format. If you don't add this header, MashZone NextGen returns the response in the native format of the mashable for mashup.
- `x-p-anonymous=true` to have MashZone NextGen treat this request as coming from a guest user with no identity.
- Headers to have MashZone NextGen paginate the response.
- Headers to override the credentials to use for a secure connection with a mashable.

  Mashables are registered with credential information when the service requires authentication, but you can override these default credentials using MashZone NextGen headers.

Using jQuery, for example, the request would look something like this:

```
var requestBody = '';
var params = "p=" + this.zip;
var headers= "x-p-resultFormat=json&x-p-anonymous=true";
var invokeUrl = "/presto/edge/api/rest/YahooWeatherREST/getData?" + headers + "&" + params;
jQuery.ajax({
  type: "get",
  contentType: "application/x-www-form-urlencoded",
  url: invokeUrl,
  data: requestBody,
  dataType: "text",
  success: function(response){
    var result = response;
    if(typeof response !== "string") {
        result = Object.toJSON(response);
    }
    jQuery("#result").val('' + result);//show response
  },
  error: function(xhr, errorStatus, e) {
      jQuery("#result").val(xhr.responseText);//handle error
  }
});
```

If the mashable or mashup uses HTTP POST, you must also build the body of the request to match what the mashable or mashup is expecting.

Cross Domain Requests

In custom apps or web pages, browser security does not allow JavaScript to send requests to servers that are not in the same domain as the web page or other scripts on the page. This is called a cross domain request.

If your app or web page will invoke mashables or mashups hosted in MashZone NextGen Servers in different domains, you must handle cross domain requests in your code.

Session Management

Session management requirements also depend on how authentication is configured in MashZone NextGen. MashZone NextGen maintains sessions for each authenticated user. This session expires if no new requests are received within the timeout period configured for MashZone NextGen.

If authentication is handled by an SSO solution or uses SSL and digital certificates, MashZone NextGen uses the user identity information from SSO or the certificate to start a new session when a request is received for a session that has expired. Similarly, for guest users, MashZone NextGen simply starts a new session.

If authentication is using simple credentials, however, MashZone NextGen returns an error message for requests using an expired session. In this case, you must handle this error appropriately.

Use MashZone NextGen Connect for Javascript

You use PC4JS to connect custom apps, web applications or web sites to the MashZone NextGen Server and invoke mashable information sources or mashups. This JavaScript API is hosted with the MashZone NextGen Server and uses the MashZone NextGen REST API.

This topic covers the basics of using PC4JS to create a connection to the MashZone NextGen Server, send a request to invoke a mashable or mashup and handle a response in two different contexts:

- "Use PC4JS in Custom Apps" on page 1484
- "Use PC4JS in Other Web Applications or Web Sites" on page 1486

Use PC4JS in Custom Apps

With custom apps, PC4JS is a dependency of the MashZone NextGen App API so it is loaded and available automatically. You do not need to add a `<require>` declaration in the App Spec. Simply "Get a Connection to the MashZone NextGen Server" on page 1484 and "Send Requests to Run Mashables or Mashups" on page 1485.

Get a Connection to the MashZone NextGen Server

Use the `getConnection()` method to get the default PC4JS connection to the MashZone NextGen Server that also hosts this app.

For example:

```
Presto.namespace("MyOrg");
var MyOrg.MyCustomApp = function(app){
  var root = jQuery( app.rootElement ).find( '#content' );
  var connection = app.getConnection();
  ...
```

```
}
```

In rare cases, you may need to get a connection to a *remote*MashZone NextGen Server (any server other than the server hosting this app). Use PC4JS to create another connection in this case, such as:

```
Presto.namespace("MyOrg");
var MyOrg.MyCustomApp = function(app){
  var root = jQuery( app.rootElement ).find( '#content' );
  var local = app.getConnection();
  var remoteUrl = "http://my.other.domain:8080/presto";
  var remote = new Presto.Connection({prestoUrl: remoteUrl});
  ...
}
```

In most cases, you do not need to login or provide user credentials for the connection. See "Guest or Authenticated Access with PC4JS" on page 1488 to determine authentication requirements.

Send Requests to Run Mashables or Mashups

Once you have a connection, you create requests to invoke MashZone NextGen mashables or mashups, supplying input parameters or optional MashZone NextGen headers as needed.

You can find a sample of the code to invoke a specific mashable or mashup in the **Technical Spec** tab on the mashable's or mashup's artifact page. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information.

You define a configuration object with properties for the request and provide callbacks for success and failure responses. A simple example would look something like this:

```
 ...
  connection.request({
      url:  "/presto/edge/api/rest/YahooWeatherREST/getData?x-presto-resultFormat=json&p=94102",
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          var result = response;
          if(typeof response !== "string") {
            result = Object.toJSON(response);
          }
           //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
</script>
```

The `url` property identifies the mashable or mashup and operation to invokeFor more information on request configuration properties, see the PC4JS Reference.

The URL can also contain input parameters for the operation and any MashZone NextGen header passed as a parameter. You can define parameter values literally or programmatically. You can also use MashZone NextGen attributes to have the MashZone NextGen Server resolve the parameter value at run time. See "Map

MashZone NextGen Attributes to Artifact Input Parameters" on page 1502 for more information.

This example also uses one very common MashZone NextGen header, `x-p-resultFormat`, as a parameter in the URL to ensure that the response is returned in JSON format. There are many other MashZone NextGen headers that your can use, however. See "MashZone NextGen Headers/Parameters" on page 1511 for a complete list.

See also "Paginate Mashable or Mashup Responses" on page 1491.

Use PC4JS in Other Web Applications or Web Sites

You can use PC4JS outside of custom apps, for other web applicatons or web sites following these steps:

1. Include the MashZone NextGenLibrary Loaderin pages that use PC4JSto invoke mashables or mashups:

   ```
   <script type="text/javascript" src="app-server:port/presto/hub/jsapi/loader.js"/>
   ```

   The Libary Loader API handles loading all of the libraries and dependencies for PC4JS and any other library installed with MashZone NextGen that you want to use.

2. Load the PC4JSlibrary using the `Presto.loadLib`method in Library Loader:

   ```
   <html>
     <head>
       <script src="http://localhost:8080/presto/hub/jsapi/loader.js"
         type="text/javascript"></script>
       <script>
         Presto.loadLib("presto-core", null, true);
   ...
       </script>
     </head>
   ...
   ```

   Use this method to load any additional configured libraries that you need. Or use other Library Loader methods to load CSS stylesheets or other JavaScript libraries.

3. Define the URL to the MashZone NextGen Serverhosting the mashables or mashups you want to work with.

   ```
   //local connection to same app server
   prestoUrl = "/presto";
   //remote connection
   var prestoUrl = "http://204.10.32.210:8080/presto";
   ```

   If you omit the URL, this defaults to the local host using the default Tomcat port (`http://localhost:8080/presto`) at runtime.

4. Create a new connection passing a configuration object including:

   - `prestoURL`: with the local or remote URL to MashZone NextGen defined previously. For example:

     ```
     var connection = new Presto.Connection({prestoUrl: prestoUrl});
     ```

- `username`*and* `password`: these properties are *only* required if the connection must be authenticated and MashZone NextGen configuration does not handle authentication automatically. See "Guest or Authenticated Access with PC4JS" on page 1488 to determine whether you need to add credentials to the connection.

  For example:

```
var connection = new Presto.Connection({prestoUrl: '/presto', username: 'aUser', password:
```

  Note: In previous releases, PC4JS allowed connections to use an *authToken* obtained from the MashZone NextGen Server when a user logged in. This is no longer needed.

5. Create a request to invoke a MashZone NextGenmashable or mashup, supplying any input parameters or optional MashZone NextGenheaders as needed.

   You can find a sample of the code to invoke a specific mashable or mashup in the **Technical Spec** tab on the mashable's or mashup's artifact page. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information.

   You define a configuration object with properties for the request and provide callbacks for success and failure responses. A simple example would look something like this:

```
<script>
  var requestBody = '';
  //local connection
  var connection = new Presto.Connection({ prestoUrl: "/presto" });
  connection.request({
      url: "/presto/edge/api/rest/YahooWeatherREST/getData?x-presto-resultFormat=json&p=94102"
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          var result = response;
          if(typeof response !== "string") {
            result = Object.toJSON(response);
          }
           //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
</script>
```

   The `url` property identifies the mashable or mashup and operation to invoke. For more information on request configuration properties, see the PC4JS Reference.

   The URL can also contain input parameters for the operation and any MashZone NextGen header passed as a parameter. You can define parameter values literally or programmatically. You can also use MashZone NextGen attributes to have the MashZone NextGen Server resolve the parameter value at run time. See "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502 for more information.

This example also uses one very common MashZone NextGen header, `x-p-resultFormat`, as a parameter in the URL to ensure that the response is returned in JSON format. You can also request results in CSV or XML format.

There are many other MashZone NextGen headers that your can use, however. See "MashZone NextGen Headers/Parameters" on page 1511 for a complete list.

See also "Paginate Mashable or Mashup Responses" on page 1491.

Guest or Authenticated Access with PC4JS

With the one exception shown below, you do not need to login or provide user credentials when sending requests with PC4JS:

If	Provide Credentials?	Why
The user is already authenticated.	No	The request has a MashZone NextGen session ID (in a cookie) that MashZone NextGen uses to determine user permissions.
MashZone NextGen is configured to use an SSO solution for authentication.	No	The SSO user agent adds user identity information to the request. MashZone NextGen does not authenticate the user but does use the identity information to determine user permissions.
MashZone NextGen is configured to use SSL and digital certificates authentication.	No	MashZone NextGen does not authenticate the user but does use identity information from the certificate to determine user permissions.
MashZone NextGen is configured to use simple credentials for authentication. This is the default authentication mechanism when you first install MashZone NextGen.	Yes	If the mashable or mashup you are invoking does *not* grant guest access, then users must have run permissions and MashZone NextGen must have user credentials to authenticate the user and check their permissions. If you omit user credentials, MashZone NextGen will process the request as a guest user causing the invocation to fail as unauthorized.

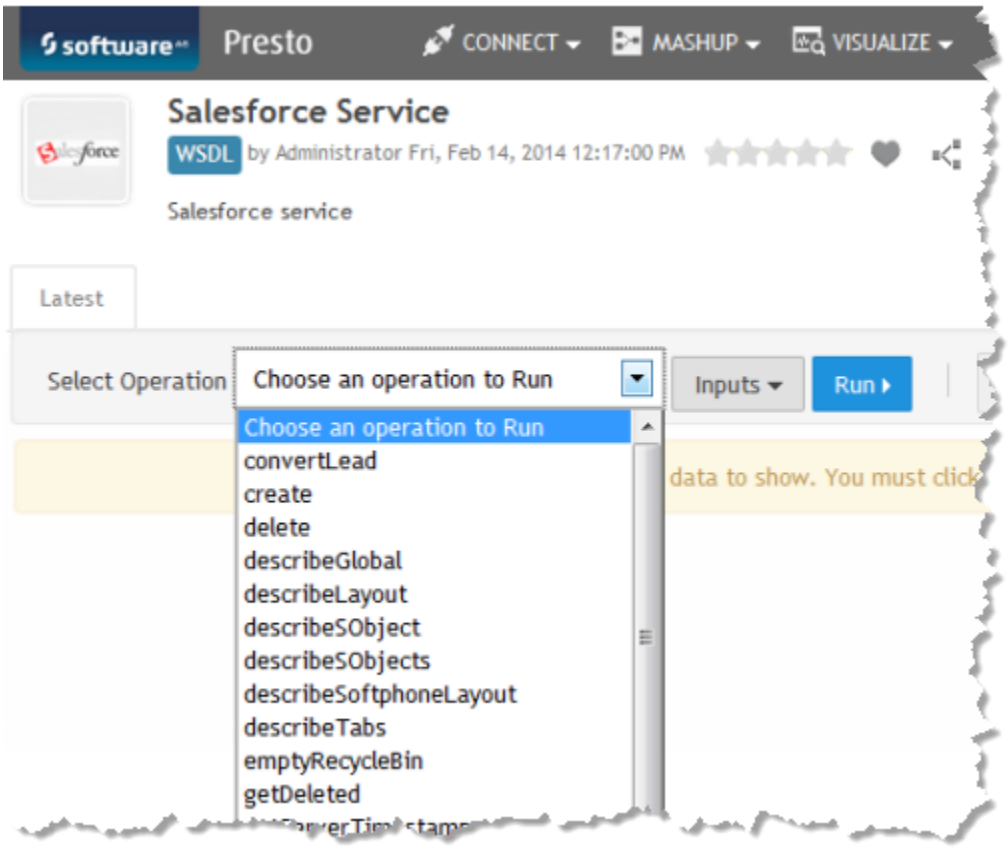
If	Provide Credentials?	Why
This request should use guest access.	No	PC4JS automatically ensures that requests with no user credentials can be processed anonymously as a guest user.

Use Mashable/Mashup Technical Specs

Most of the information you need to invoke mashables or mashups programmatically using either the MashZone NextGen REST API or MashZone NextGen Connect for JavaScript (PC4JS) API is available as a **Technical Spec** on each mashable's or mashup's artifact page.

To find this information:

1. Find the mashable or mashup from **Search** in the MashZone NextGen Hub main menu or from other links and open the artifact page for that mashable or mashup.
2. If this is a mashable and it has more than one operation, choose the operation you want to invoke.

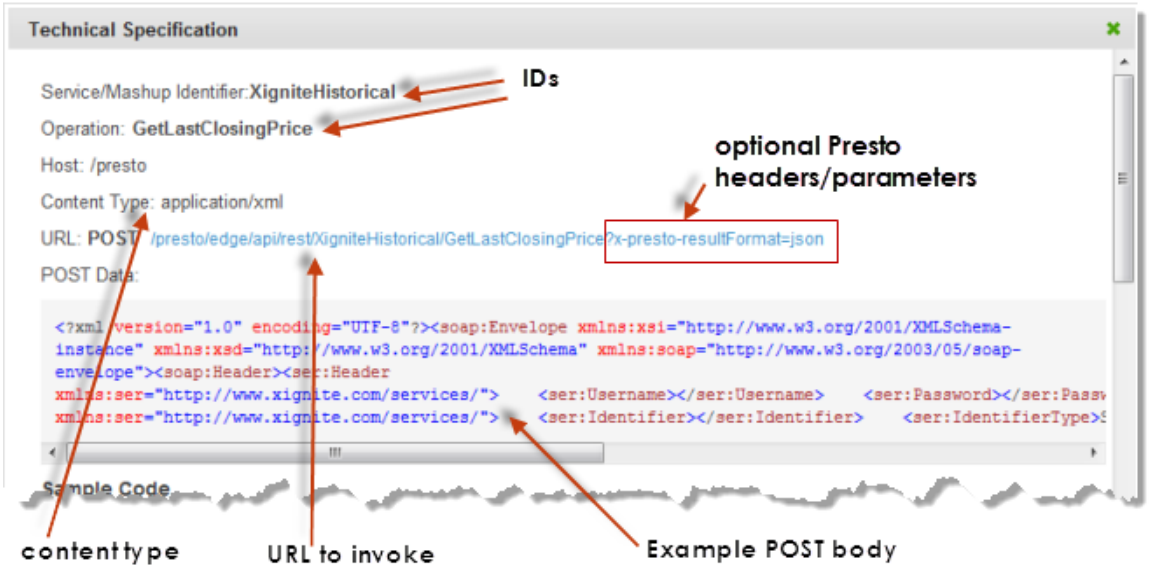


You can also change the input parameter values, if desired.

3. Select **Show >** **Technical Spec**. The specific details, including the REST URL to use and the content type are shown:

- To see raw mashable or mashup results in JSON format, right-click the **URL** and open this in another tab. This is the default format because of the `x-p-resultFormat=json` parameter.
- To export results as CSV:
  - Right-click the **URL** and open this in another tab.
  - Edit the URL in the address bar and change the parameter to `x-p-resultFormat=csv`.
  - Save the results as a file when prompted.
- You can change any of the values for parameters or add or delete any "MashZone NextGen Headers/Parameters" on page 1511 to the URL, if needed.

If the mashable or mashup requires a POST request, this also includes a sample of the POST body:



4. If you are working in JavaScript, you can also simply copy and tweak any of the code samples shown:

This includes samples using MashZone NextGen Connect for JavaScript (PC4JS), the MashZone NextGen DataTable API or jQuery.

Paginate Mashable or Mashup Responses

MashZone NextGen supports server caching and pagination for clients using "MashZone NextGen Headers/Parameters" on page 1511 in requests to invoke mashables or mashups and in the responses containing mashable or mashup results.

In addition to the basic headers used in pagination, you can also use the MashZone NextGen DataTable APIas a client data model to simplify pagination overhead.

Note: In previous releases, MashZone NextGen supported server pagination only for mashups. This limitation no longer applies.

See "MashZone NextGen Headers Used in Pagination" on page 1492 for a basic understanding of the request and response headers used in pagination. For an example of a simple app that uses pagination, see "Pagination Using the MashZone NextGen DataTable API" on page 1492.

MashZone NextGen Headers Used in Pagination

Pagination requests and responses follow a simple pattern:

1. Initial or new request that invokes a mashable or mashup and requests pagination from the MashZone NextGen Server. These requests use the following MashZone NextGen Headers or Parameters:
   - `x-p-paginateRecord` = identifies the repeating items in the mashable or mashup results that should be paginated.
   - `x-p-paginateStart` = indicates the starting record number to return or an initial invocation to cache.
   - `x-p-paginateCount` = the number of repeating items to return in as one page.
2. The response to any request that involves pagination which includes the response with the specific repeating items requested, plus these headers with pagination information:
   - `x-paginateToken` = a token identifying the cached response for this pagination set.
   - `x-totalCount` = the total number of repeating items in the cached response.

   Note: If the pagination record sent in the request has no matches, the full response from the mashable is returned.
3. Subsequent requests for specific paginated sets of records from the cached response. These requests include the same three headers as initial requests, *plus*`x-paginateToken` as a header to identify the cached response to use.

For more information on these headers, see "MashZone NextGen Headers/Parameters" on page 1511.

Pagination Using the MashZone NextGen DataTable API

The MashZone NextGen DataTable API allows you to easily send requests and handle responses that require pagination. This API also provides a simple client data model for responses.

1. Download a template for a new custom app named `SampleDataTblPagination`. If needed, see "Create Custom Apps from the Base App Package" on page 1226 for instructions.

   Unpack the template Zip file under the web applications home for the application server hosting your MashZone NextGen Server. If you are using the default Tomcat server for MashZone NextGen, this would be the /MashZone NextGen*version* / appserver/apache-tomcat-6.0.29/webapps folder.

2. Add Pagination Buttons in the App HTML
3. Add CSS Styles for the Pagination Buttons
4. Add the DataTable Library to the App Specification
5. Start a Constructor for the Custom App
6. Create the App DataTable
7. Add Configuration and Callbacks for the DataTable
8. Invoke the Mashable, Load the DataTable and Render the App
9. Add a Listener to the Load Event
10. Add Handlers for the Pagination Buttons

Add Pagination Buttons in the App HTML

Open SampleDataTblPagination/html/app.html in the editor of your choice and update the HTML tags.

```
<div
>
  <div
>
    <button id="firstItem">&lt;&lt;</button>
    <button id="prevItem">&lt;</button>
    <button id="nextItem">&gt;</button>
    <button id="lastItem">&gt;&gt;</button>
  </div>
  <div>
    <table
>
     <thead>
      <tr>
        <th>Title</th>
        <th>Address</th>
        <th>City</th>
        <th>Phone</th>
        <th>Distance</th>
        <th>URL</th>
      </tr>
     </thead>
     <tbody
></tbody>
    </table>
  </div>
</div>
```

Add CSS Styles for the Pagination Buttons

Open the SampleDataTblPagination/css/app.css stylesheet and add the appropriate styles for the table and buttons:

```
div.sampleTbl { font-family: Arial, Verdana, sans-serif; font-size: 9pt;
  display: block;}
table.myTable {border: 1px solid #c9c9c9; border-collapse: collapse;
  margin-top: 10px;}
table.myTable thead {background-color: #ededed;}
table.myTable th, table.myTable td {padding: 5px; border: 1px solid #c9c9c9;}
.buttons {padding: 5px; display:block; }
.buttons button{ margin:0 4px 0 0; padding: 4px; background-color:#ededed;
  border:1px solid #c9c9c9; border-top:1px solid #cbcbcb;
  border-left:1px solid #cbcbcb; color: #666; font-size:100%; line-height:130%;
  text-decoration:none; font-weight:bold; cursor:pointer; width:auto;
  overflow:visible; }
```

Add the DataTable Library to the App Specification

Next, we must update the App Specification to add the JavaScript libraries it needs and use the correct constructor for the app.

Open SampleDataTblPagination/app.xml in the editor of your choice. The initial App Specification is based on the sample Hello World app shipped with MashZone NextGen. First, let's update the basic information about the app such as the `<title>` and `<description>`:

```
<app id="SampleDataTblPagination" name="SampleDataTblPagination"
   jsclass="Sample.HelloWorld" height="200" width="200"
   draggable="false" minimizable="false">
  <title>SampleDataTblPagination</title>
  <description>Sample to invoke one mashable using the DataTable API
to enable server pagination and render results in a simple table
with a pagination bar. </description>
  ...
```

Next, we update the `jsclass` attribute on `<app>` to match the name of the JavaScript class we will write for this custom app. Change this to `Sample.DataTablePagination`.

```
<app id="SampleDataTblPagination" name="SampleDataTblPagination"
   jsclass="Sample.DataTablePagination" height="200" width="200"
   draggable="false" minimizable="false">
  <title>SampleDataTblPagination</title>
  ...
```

Last, the app needs to work with both the MashZone NextGen DataTable API and the JQuery Templating API. To make sure these JavaScript libraries are loaded, add two `<require>` elements to the `<requires>` section, like this:

```
<app id="SampleDataTblPagination" name="SampleDataTblPagination"
   jsclass="Sample.DataTablePagination" height="200" width="200"
   draggable="false" minimizable="false">
  <title>SampleDataTbl</title>
  ...
  <requires>
    <require name="presto-data" type="library" version="1.0"/>
    <require name="jquery-tmpl" type="library" version="1.0"/>
    <require src="js/app.js" type="script"/>
    <require src="css/app.css" type="css"/>
    <require src="html/app.html" type="html"/>
```

```
  </requires>
</app>
```

Start a Constructor for the Custom App

The behavior for the custom app is defined in the app.js library, in this case in SampleDataTblPagination/js/app.js.

Open app.js in the editor of your choice to create the constructor for the app. Then declare a namespace for the app and start the constructor function:

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
};
```

App constructors should always get a reference to the root node for the app. Apps that invoke MashZone NextGen mashables or mashups should also get a reference to the default connection to the MashZone NextGen Server:

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
  var rootDiv = jQuery(app.getRootElement);
  var connection = app.getConnection();
 };
```

For this app, we also need to define a JQuery template to use to generate the rows of the table once the app receives a response. This typically involves adding HTML tags and templating syntax to map fields in the response:

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
  var rootDiv = jQuery(app.getRootElement);
  var connection = app.getConnection();
  var rowMarkup = "<tr><td>${Title}</td><td>${Address}</td><td>${City}</td>
<td>${Phone}</td><td>${Distance}</td><td>${BusinessUrl}</td></tr>"
  jQuery.template("rowTemplate", rowMarkup);
 };
```

Create the App DataTable

This example app will use a DataTable to send a request to the MashZone NextGen Server to invoke the Yahoo local Search mashable, and once the result is cached, to request other pages from cache. The DataTable acts as a client data model, providing some utility methods and simplifying the process of working with the paginated results.

To create a DataTable, we need:

- Some variables to track pagination status
- Configuration information for the request to send to the MashZone NextGen Server
- A connection to the MashZone NextGen Server

The variables we need to track include:

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
  var rootDiv = jQuery(app.getRootElement);
  var connection = app.getConnection();
  var rowMarkup = "<tr><td>${Title}</td><td>${Address}</td><td>${City}</td>
<td>${Phone}</td><td>${Distance}</td><td>${BusinessUrl}</td></tr>"
```

```
  jQuery.template("rowTemplate", rowMarkup);
  var cursor = 1;  //first item in current page
  var total = 0;   //total number of items cached in server
  var totalPages = 0; // number of pages availables
  var pageSize = 5; // number of rows per page
 };
```

The configuration information needed to send requests includes the URL to invoke the mashable, the HTTP method to use, the content type, and for POST requests, the body of the request.

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
...
  var cursor = 1;  //first item in current page
  var total = 0;   //total number of items cached in server
  var totalPages = 0; // number of pages availables
  var pageSize = 5; // number of rows per page
  requestBody = '';
  var reqConfig = { uri:  "/presto/edge/api/rest/YahooLocalSearchREST/getData?x-presto-resultForm
      .kcC72DV34FYTpAGuwwbV8YGI.DsMBQ0RB9eZARS621ecnHq33c.g1XJV93a64hrdaM3&query=banks&zip=94102&
        type: "get",
        contentType: "application/x-www-form-urlencoded",
        data: requestBody
      };
 };
```

You can get this information from the Technical Specs tab in the artifact page for any mashable or mashup. If needed, see "Use Mashable/Mashup Technical Specs" on page 1489 for more information.

Last, we must define the "record" to be paginated. This is defined with an XPath expression that identifies the repeating item in the results. Once we have the record defined, we create a new DataTable for the app:

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
...
  var cursor = 1;  //first item in current page
  var total = 0;   //total number of items cached in server
  var totalPages = 0; // number of pages availables
  var pageSize = 5; // number of rows per page
  requestBody = '';
  var reqConfig = { uri:  "/presto/edge/api/rest/YahooLocalSearchREST/getData?x-presto-resultForm
      json&appid=.kcC72DV34FYTpAGuwwbV8YGI.DsMBQ0RB9eZARS621ecnHq33c.g1XJV93a64hrdaM3&query=banks
        type: "get",
        contentType: "application/x-www-form-urlencoded",
        data: requestBody
      };
  var record = "/ResultSet/Result";
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
 };
```

Add Configuration and Callbacks for the DataTable

You use the `load(params,options,callbacks)` method in DataTable to send requests to the MashZone NextGen Server and populate the DataTable. In this case,

the parameters to invoke the mashable are already defined when the DataTable was instantiated. You can use this to override default parameters, however:

```
...
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
};
```

The *options* parameter is a configuration object that typically contains pagination configuration. Pagination configuration must define the record to be paginated, the starting record to return and how many records to return, like this:

```
...
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
  options.record = record;
  options.start = cursor;
  options.count = pageSize;
};
```

The last parameter is an object that must contain two callback functions: `onSuccess` and `onFailure`, such as this:

```
...
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
  options.record = record;
  options.start = cursor;
  options.count = pageSize;
  var cb = { onSuccess: function(rows, response, headers, options) {},
             onFailure: function(e) {}
           };
};
```

The DataTable is populated by successful results, so `onSuccess` must render the rows of the table with the set of records returned in the response. If the response has the appropriate results, the `cursor` variable must be updated to point to the first record in this page set.

```
...
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
  options.record = record;
  options.start = cursor;
  options.count = pageSize;
  var cb = { onSuccess: function(rows, response, headers, options) {
               if (response.ResultSet.Result) {
```

```
                    cursor = options.start;
                    var locations = rows;
                    jQuery('.tblBdy').empty();
                    jQuery.tmpl("rowTemplate", locations).appendTo(".tblBdy");
                  } else {
                    rootDiv.html("no results found");
                }
              },
              onFailure: function(e) {}
            };
};
```

The `rows` object contains an array of objects, each with properties for the fields in that record. We use this, with the JQuery template to render the rows of the table for the app and append them to the table body.

Last, we add some simple error handling to `onFailure`:

```
...
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
  options.record = record;
  options.start = cursor;
  options.count = pageSize;
  var cb = { onSuccess: function(rows, response, headers, options) {
               if (response.ResultSet.Result) {
                    cursor = options.start;
                    var locations = rows;
                    jQuery('.tblBdy').empty();
                    jQuery.tmpl("rowTemplate", locations).appendTo(".tblBdy");
                  } else {
                    rootDiv.html("no results found");
                }
              },
              onFailure: function(e) {
                    rootDiv.html(e.message);
                }
            };
};
```

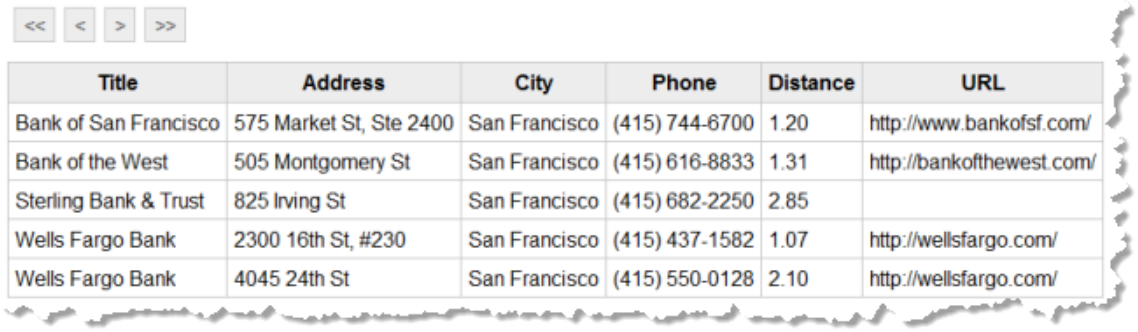
Invoke the Mashable, Load the DataTable and Render the App

Once the pagination configuration and callbacks are defined, you invoke the mashable, populate the DataTable and render the app using the `load` method, as shown here:

```
...
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
  options.record = record;
  options.start = cursor;
  options.count = pageSize;
  var cb = { onSuccess: function(rows, response, headers, options) {
               if (response.ResultSet.Result) {
                    cursor = options.start;
                    var locations = rows;
```

```
                    jQuery('.tblBdy').empty();
                    jQuery.tmpl("rowTemplate", locations).appendTo(".tblBdy");
                  } else {
                    rootDiv.html("no results found");
                }
              },
              onFailure: function(e) {
                    rootDiv.html(e.message);
              }
            };
  dataTable.load(params, options, cb);
};
```

If you save your changes to the custom app JavaScript library, you can test the app, at this point. Login to MashZone NextGen in the browser you want to test with. Then open another tab and open http://*app-server:port* /SampleDataTblPagination/index.html for your application server. You should see something like this:



Title	Address	City	Phone	Distance	URL
Bank of San Francisco	575 Market St, Ste 2400	San Francisco	(415) 744-6700	1.20	http://www.bankofsf.com/
Bank of the West	505 Montgomery St	San Francisco	(415) 616-8833	1.31	http://bankofthewest.com/
Sterling Bank & Trust	825 Irving St	San Francisco	(415) 682-2250	2.85	
Wells Fargo Bank	2300 16th St, #230	San Francisco	(415) 437-1582	1.07	http://wellsfargo.com/
Wells Fargo Bank	4045 24th St	San Francisco	(415) 550-0128	2.10	http://wellsfargo.com/

Only the first five results display in the table. We need to fix the pager bar to allow users to see any remaining results.

Add a Listener to the Load Event

To allow the pager buttons to function properly, the app needs to know the total number of records that are cached. To keep track of this, we edit SampleDataTblPagination/js/app.js and add a listener to the DataTable for the `load` event. DataTable fires three events, `beforeload`, `load` and `loaderror`, that apps can use.

Note: DataTable extends Presto.Observable which provides methods to listen to and fire events.

Once the DataTable is populated, we can use the `getTotalCount()` method to determine how many records are cached in the MashZone NextGen Server for this invocation and to calculate the total number of pages to support:

```
...
  var cb = { onSuccess: function(rows, response, headers, options) {
              if (response.ResultSet.Result) {
                cursor = options.start;
                var locations = rows;
                jQuery('.tblBdy').empty();
                jQuery.tmpl("rowTemplate", locations).appendTo(".tblBdy");
              } else {
                rootDiv.html("no results found");
              }
```

```
            },
            onFailure: function(e) {
                rootDiv.html(e.message);
            }
          };
 dataTable.addListener("load", function(rows) {
          total = dataTable.getTotalCount();
          totalPages = parseInt(total / pageSize,10);
  });
  dataTable.load(params, options, cb);
};
```

Add Handlers for the Pagination Buttons

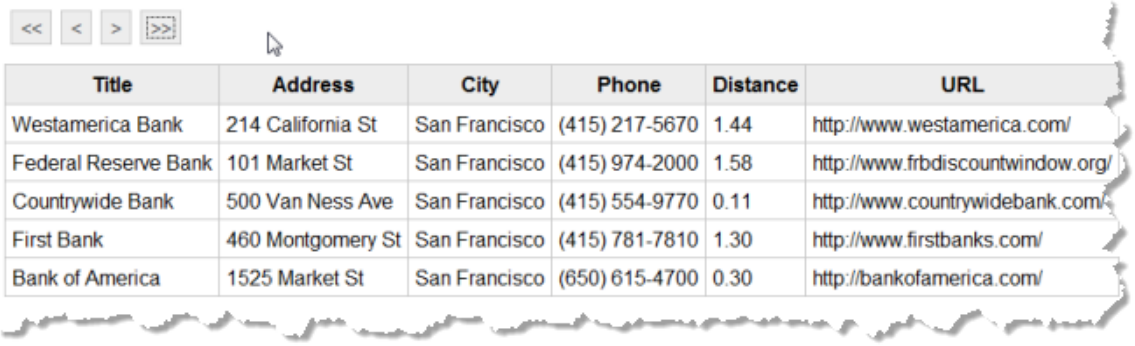
Now that the full page state is known, we are ready to add handler functions for the pagination buttons. These use the `page(`*`options`*`, `*`callback`*`)` method for the DataTable to request the appropriate page set.

The *`options`* parameter contains the `start` and `count` properties to request the specific set of records for a given page. The *`callback`* parameter can use the same callbacks as the `load()` method.

To do this, we use JQuery to attach event handlers to the appropriate buttons:

```
...
 dataTable.addListener("load", function(rows) {
          total = dataTable.getTotalCount();
          totalPages = parseInt(total / pageSize,10);
  });
  jQuery('#firstItem').click( function(){
    options.start = 1;
    options.count = 5;
    dataTable.page(options, cb);
  });
  jQuery('#prevItem').click( function(){
    cursor = options.start;
    if (cursor > 5) {
      options.start = cursor - 5;
    } else {
      options.start = 1;
    }
    dataTable.page(options, cb);
  });
  jQuery('#nextItem').click( function(){
    cursor = options.start;
    if (total > 0 && cursor + 5 < total ) {
      options.start = cursor + 5
    }
    dataTable.page(options, cb);
  });
  jQuery('#lastItem').click( function(){
    options.start = (totalPages-1)*pageSize + 1;
    dataTable.page(options, cb);
  });
  dataTable.load(params, options, cb);
};
```

If you save these changes and test the app once more, you should now be able to use the pager buttons to page through the entire set of results:



<< < > >>

Title	Address	City	Phone	Distance	URL
Westamerica Bank	214 California St	San Francisco	(415) 217-5670	1.44	http://www.westamerica.com/
Federal Reserve Bank	101 Market St	San Francisco	(415) 974-2000	1.58	http://www.frbdiscountwindow.org/
Countrywide Bank	500 Van Ness Ave	San Francisco	(415) 554-9770	0.11	http://www.countrywidebank.com/
First Bank	460 Montgomery St	San Francisco	(415) 781-7810	1.30	http://www.firstbanks.com/
Bank of America	1525 Market St	San Francisco	(650) 615-4700	0.30	http://bankofamerica.com/

The complete class for this custom app is:

```
Presto.namespace("Sample");
Sample.DataTablePagination = function( app ) {
  var rootDiv = jQuery(app.getRootElement);
  var connection = app.getConnection();
  var rowMarkup = "<tr><td>${Title}</td><td>${Address}</td><td>${City}</td>
<td>${Phone}</td><td>${Distance}</td><td>${BusinessUrl}</td></tr>"
  jQuery.template("rowTemplate", rowMarkup);
  var cursor = 1;  //first item in current page
  var total = 0;   //total number of items cached in server
  var totalPages = 0; // number of pages availables
  var pageSize = 5; // number of rows per page
  requestBody = '';
  var reqConfig = { uri:  "/presto/edge/api/rest/YahooLocalSearchREST/getData?x-presto-resultForm
   json&appid=.kcC72DV34FYTpAGuwwbV8YGI.DsMBQ0RB9eZARS621ecnHq33c.g1XJV93a64hrdaM3&query=banks&zi
        type: "get",
        contentType: "application/x-www-form-urlencoded",
        data: requestBody
      };
  var record = "/ResultSet/Result";
  var dataTable = new Presto.DataTable( {
    request: reqConfig,
    connection: connection,
    record: record
    });
  var params = {};
  options.record = record;
  options.start = cursor;
  options.count = pageSize;
  var cb = { onSuccess: function(rows, response, headers, options) {
              if (response.ResultSet.Result) {
                cursor = options.start;
                var locations = rows;
                jQuery('.tblBdy').empty();
                jQuery.tmpl("rowTemplate", locations).appendTo(".tblBdy");
              } else {
                rootDiv.html("no results found");
              }
            },
            onFailure: function(e) {
                rootDiv.html(e.message);
            }
          };
 dataTable.addListener("load", function(rows) {
          total = dataTable.getTotalCount();
          totalPages = parseInt(total / pageSize,10);
  });
```

```
  jQuery('#firstItem').click( function(){
    options.start = 1;
    options.count = 5;
    dataTable.page(options, cb);
  });
  jQuery('#prevItem').click( function(){
    cursor = options.start;
    if (cursor > 5) {
      options.start = cursor - 5;
    } else {
      options.start = 1;
    }
    dataTable.page(options, cb);
  });
  jQuery('#nextItem').click( function(){
    cursor = options.start;
    if (total > 0 && cursor + 5 < total ) {
      options.start = cursor + 5
    }
    dataTable.page(options, cb);
  });
  jQuery('#lastItem').click( function(){
    options.start = (totalPages-1)*pageSize + 1;
    dataTable.page(options, cb);
  });
  dataTable.load(params, options, cb);
};
```

Map MashZone NextGen Attributes to Artifact Input Parameters

You can have the MashZone NextGen Server resolve input parameters for apps, mashables or mashups at runtime from server data, called MashZone NextGen attributes, rather than providing this directly in an app or other client. This is also sometimes called *server-side binding* or *server templating*.

Common use cases for attribute mapping include:

- Information for the current user, such as credentials for a web service.
- Shared information that should be used for all contexts, such as an application ID or other key that should always be used when invoking a specific mashable.
- Information provided in a previous response that is specific to the current transaction or session.

The syntax to map artifact input parameters to MashZone NextGen attributes is different for different contexts:

- Parameters specified in the URL to run the artifact use a *mapping expression*.
- EMML statements in a mashup declare attributes as variables or input parameters and then refer to these variables just like any other variable. See "Using MashZone NextGen Attributes in Mashups" on page 614 for more information.

MashZone NextGen Attribute Mapping Expressions

Scope	Mapping Expressions for URLs	Variable or Parameter Declarations in EMML
MashZone NextGenglobal attributes: custom attributes with simple values used for: ■ All users. ■ Default values for MashZone NextGen user attributes. If a value is not available at runtime for the current user, the MashZone NextGen global attribute value is used.	*parameter* = `$global/`*attribute-name*	`<variable name="global.`*attribute-name*`" type="`*simple-type*`"/>`
MashZone NextGenuser attributes: custom attributes with simple values for each specific user. In addition to the custom attributes that you define, MashZone NextGen user attributes can also refer to user attributes defined in your User Repository. Global attributes can provide default values for user attributes with the same name.	*parameter* = `$user/`*attribute-name*	`<variable name="user.`*attribute-name*`" type="`*simple-type*`"/>`
MashZone NextGensession attributes: simple values or complex objects with multiple properties.	*parameter* = `$session/`*attribute-name* `[/`*Xpath-expression-to-a-specific-node* `]`	For session attributes with simple values: `<variable name="session.`*attribute-*

Scope	Mapping Expressions for URLs	Variable or Parameter Declarations in EMML
They are data saved from a previous response during the user's current session. To use session attributes, your app or other client must first define them with the MashZone NextGen Server. See "Define MashZone NextGen Session Attributes" on page 1505 for instructions. **Important:** Effective in this release, the user's MashZone NextGen credentials are *no longer* available as session attributes.	The XPath expression is optional and is only relevant for session attributess that are complex objects.	*name*" type="*simple-type*"/> For complex objects: <variable name="session.*attribute-name*" type="document"/>
MashZone NextGenrequest attributes: refers to HTTP headers or cookies for the current request.	*parameter* = $request.*http-header/cookie-name*	MashZone NextGen request attributes are not supported in EMML.

The following example shows a MashZone NextGen attribute for an app using PC4JS that maps a video service key to a global attribute:

Global Attribute Mapping

```
Presto.namespace("MyOrg");
 var MyOrg.MyVideosApp = function(app){
 var root = jQuery( app.rootElement ).find( '#content' );
 var connection = app.getConnection();
 var requestBody = "";
 connection.request({
     url:  "/presto/edge/api/rest/VideoSvc/getAll?x-presto-resultFormat=json&videoKey=$global.Vi
     type: "get",
     contentType: "application/x-www-form-urlencoded",
     data: requestBody
   },
   { onSuccess: function(response, responseHeaders) {
         //handle response data
       },
```

```
      onFailure: function(e) {
          //handle error message
    }
  });
```

This next example show an app that maps an input parameter for the user's email to a MashZone NextGen user attribute.

User Attribute Mapping

```
...
  connection.request({
      url:  "/presto/edge/api/rest/myMashup/Invoke?x-presto-resultFormat=json&email=$user.email",
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
```

The next example shows an app that maps an input parameter to a cookie in the current request:

Request Attribute Mapping

```
...
  connection.request({
      url:  "/presto/edge/api/rest/myMashup/Invoke?x-presto-resultFormat=json&ssocookie=$request.
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
```

See "Define MashZone NextGen Session Attributes" on page 1505 for examples of defining and mapping MashZone NextGen session attributes.

Define MashZone NextGen Session Attributes

You define MashZone NextGen session attributes in the request that generates the response with the data to be saved. Session attributes allow you to save data from one request and use it in subsequent requests for the current session. It is useful for sensitive or personal information that should remain in the MashZone NextGen Server for security or privacy reasons.

You use the optional MashZone NextGen Header/Parameter *x-p-sessionbinding* to define session attributes. This header takes an object with attribute definition expressions. Each

expression provides the name for a new session attribute and an XPath expression that identifies the node within the response that should be saved in this attribute for the current session.

The XPath expression can identify a node with a simple value, such as `'@session.ssn':'/customer/SSN'`.

They can also identify nodes sets, such as `'@session.articles':'/rss/channel/item'` or one node with complex content, such as `'@session.firstArticle':'/rss/channel/item[1]'`. See "A Brief Introduction to XPath 2.0" on page 596 for more information on XPath expressions.

Note: The @ symbol defines a session attribute. You refer to session parameters with $, just as you do with global or user parameters. See "Define MashZone NextGen Session Attributes" on page 1505 for this syntax.

This first session mapping example takes a query response and maps two nodes with simple content to two MashZone NextGen session attributes.

Defining Session Attributes with Simple Values

```
...
  connection.request({
      url:  "/presto/edge/api/rest/CustomerQuery/getCustomer?id=mr321y98&x-presto-resultFormat=
           json&x-p-sessionbinding={'@session.ssn':'/customer/SSN','@session.name':'customer/name
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
```

A later request then maps these MashZone NextGen session attributes to mashable input parameters:

Mapping Session Attributes with Simple Values

```
...
  connection.request({
      url:  "/presto/edge/api/rest/CustomerNotices/renewalNotice?x-presto-resultFormat=
            json&custSSN=$session/ssn&custName=$session/name,
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
```

The next session mapping example shows a mapping from an RSS feed. This defines a complex MashZone NextGen session attribute that contains feed data for the first article from this web feed.

Defining Session Attributes with Complex Content

```
...
  connection.request({
      url:  "/presto/edge/api/rest/someRSSFeed/getData?x-presto-resultFormat=json&x-p-sessionbind
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
```

A later request then maps data from nodes within this MashZone NextGen session attribute to input parameters:

Mapping Session Attributes with Complex Content

```
...
  connection.request({
      url:  "/presto/edge/api/rest/myMayshup/Invoke?x-presto-resultFormat=json&article=$session/
             firstArticle/item/title&link=$session/firstArticle/rss/channel/item/link,
      type: "get",
      contentType: "application/x-www-form-urlencoded",
      data: requestBody
    },
    { onSuccess: function(response, responseHeaders) {
          //handle response data
        },
      onFailure: function(e) {
          //handle error message
    }
  });
```

MashZone NextGen APIs

MashZone NextGen has several APIs that you can use to connect to MashZone NextGen, to create pluggable views or custom apps or to extend MashZone NextGen features. Reference documentation for these APIs is available from the following links.

MashZone NextGen REST APIs and MashZone NextGen Connect

Applications can use the following APIs to connect to the MashZone NextGen Server and invoke MashZone NextGen mashables or mashups:

- "MashZone NextGen REST API" on page 1509

- MashZone NextGen Connect for JavaScript
- "MashZone NextGen Snapshot API" on page 1517

See also "MashZone NextGen Headers/Parameters" on page 1511 that can be used with these APIs.

Pluggable Views, Custom Apps and Common JavaScript Client Libraries

When the MashZone NextGen built-in views do not provide the visualizations you need, you can add pluggable views to the MashZone NextGen View Gallery and the MashZone NextGenView Maker using the MashZone NextGen Views API.

Pluggable views use a *datatable* from the MashZone NextGen DataTable APIMashZone NextGen DataTable API as the interface to mashable or mashup results. This API provides a light-weight, tabular client-side data model that you can also use in custom apps to work with mashable or mashup results.

When the basic apps generated by App Maker do not provide needed functionality, you can use the MashZone NextGen App API in conjunction with the App Specifications to define custom apps.

The MashZone NextGen Library Loader API provides methods to easily load JavaScript or CSS libraries automatically and seamlessly handle any dependencies. Both the MashZone NextGen View Framework and the MashZone NextGen Apps Framework use the Library Loader to load dependencies for view or apps, including third-party libraries that are bundled in MashZone NextGen and any pluggable libraries that you add to MashZone NextGen. Typically, pluggable libraries include code from your organization or other third parties that you provide to extend base functionality, such as libraries for pluggable views.

Miscellaneous Extension APIs

Extension APIs for MashZone NextGen provide extension points to accommodate application requirements. These include:

- Custom XPath Functions for Mashup Scripts.

  This API allows you to define custom XPath functions to perform calculations or transformations within XPath expressions in mashup scripts.

- User-Defined Analytic Functions for large dataset queries with RAQL.

  This API allows you to define window analytic function or aggregate analytic functions in addition to the built-in MashZone NextGen functions that you can use in RAQL queries in mashup scripts. You can also define plain functions for RAQL queries which do not require this API.

- Custom SSO Filters for MashZone NextGen Authentication with Single Sign-On solutsions.

This API allows you to extend the default identity extraction behavior of MashZone NextGen when authentication is integrated with a Single Sign-On solution.

- Custom Certificate Validation for MashZone NextGen Authentication with Digital Certificates.

  This API allows you to extend the default certification validation behavior of MashZone NextGen when authentication uses digital certificates.

- Custom Security Profiles for mashable information sources.

  MashZone NextGen provides security profiles for the most common types of secure connections including basic authentication, NTLM, SSL or single sign-on. This API allows you to define and handle custom authentication or secure connection requirements with mashable information sources.

MashZone NextGen REST API

The MashZone NextGen REST API supports HTTP GET and POST requests. The MashZone NextGen API does *not* support HTTP PUT or DELETE requests.

Requests may be anonymous (a guest user with no identity) or authenticated, but are limited to those actions permitted to guests or the specific user. See "Handle Authentication" on page 1509 for details. You can "Invoke Mashables or Mashups" on page 1510 or work with snapshots (see "MashZone NextGen Snapshot API" on page 1517 for details).

Handle Authentication

Requests can be *anonymous* to have MashZone NextGen treat the user as a guest. To send anonymous requests, include the `x-p-anonymous`MashZone NextGen header/parameter in the URL or as an HTTP header.

In all other cases, the first request to MashZone NextGen must supply user identity information. This creates a standard HTTP session. What is required in the initial request depends on how authentication is configured in MashZone NextGen.

For authentication with SSO (single sign-on solution) or SSL with digital certificates, user identity information is supplied by those solutions. No additional authentication information is required in the request.

For authentication using basic credentials, you must use the REST API method to login to create a MashZone NextGen session and authenticate the user. This adds a cookie with session information which is then included in all subsequent requests. You can also use the REST API method to logout and end the session or simply let the session expire (based on MashZone NextGen timeout configuration).

- *Login*: use this URL:

  `http://`*app-server*`:`*port*`/presto/edge/api/rest/UserManagerService/login?presto_username=`*username*`&presto_password=`*pw*

Both parameters must be in clear text.

Successful logins set a session cookie and return an XML response in the form:

```
<?xml version="1.0"?>
<ServiceResponseObject>
  <version>2.0</version>
  <appId></appId>
  <sid>UserManagerService</sid>
  <svcVersion>1.0</svcVersion>
  <oid>login</oid>
  <response
>
    <authToken>Token is set as HTTP cookie</authToken>
  </response>
  <errorCode></errorCode>
  <invId></invId>
  <header/>
  <serviceHeaders/>
</ServiceResponseObject>
```

Unsuccessful logins return a simple text response: `Authentication failed.Userid:someName`.

- *Logout*: use this URL:

  `http://`*app-server* `:`*port* `/presto/edge/api/rest/UserManagerService/logout`

Invoke Mashables or Mashups

The form for URLs to invoke MashZone NextGen mashables or mashups using the REST API is:

`http://`*app-server* `:`*port* `/presto/edge/api/rest/`*mashable-mashup-id* `/`*operation-id* `[?`*input-parameters-if-any* `&`*presto-parameters-if-any*

- *mashable-mashup-id* = the ID of the mashable or mashup to run. See "Use Mashable/Mashup Technical Specs" on page 1489 for information on obtaining artifact IDs.
- *operation-id* = the ID of the operation in this mashable or mashup to run.

  Many mashables, such as syndicated web feeds, have a single operation. Mashups also have a single operation.
- *input-parameters-if-any* = add input parameters as query parameters in the URL when the mashable or mashup uses the HTTP GET action.

  `paramA=valueA&paramB=valueB`
- *presto-parameters-if-any* = optionally, add any "MashZone NextGen Headers/Parameters" on page 1511 to the URL to control processing, request specific information in the response or pass additional information to a mashable. Some of the most common MashZone NextGen parameters include:
  - `x-p-anonymous` to have this request treated as a guest user, with no authentication required.

- `x-p-resultFormat` to have MashZone NextGen return the response in CSV, JSON or XML formats.
- `x-p-paginateStart` and other pagination parameters to have MashZone NextGen paginate the result and return one 'page' of data.

See "Working with theMashZone NextGenREST API" on page 1480 for a basic example of using this API.

MashZone NextGen Headers/Parameters

MashZone NextGen provides custom headers/parameters to enable you to:

- Control certain aspects of request processing
- Provide information directly to a mashable
- Override defaults such as credentials for mashables
- Request specific information in the response

These processing flags can be added to requests as either HTTP headers or as parameters in a request URL. You can also add these processing flags in JUMP requests. See "MashZone NextGen Request Headers/Parameters" on page 1511 for a complete list.

Requests to run apps may also include some app-specific parameters. See "Parameters for App URLs" on page 1294 for a list.

MashZone NextGen may also return specific information as HTTP headers in responses. See "MashZone NextGen Response Headers" on page 1516 for a complete list.

Note: In version 2.7 and earlier, MashZone NextGen header/parameters were JUMP headers. JUMP is supported in 3.0 and above, although it has been deprecated. Some JUMP headers are no longer supported in versions 3.0 and above. See "Obsolete JUMP Headers" on page 1517 for more information.

MashZone NextGen Request Headers/Parameters

Request headers can be sent as HTTP headers or as parameters in request URLs via the REST API or using MashZone NextGen Connect for JavaScript (PC4JS). You may use `x-presto-`*headerName* or `x-p-`*headerName* synonymously.

Note: Some headers are only applicable to APIs or requests using the JUMP protocol.

presto_password	The password for basic credentials for this request. The value must be passed in clear text. Typically, this header should only be used, in combination with the `presto_username` header, to log in or authenticate a new principal when

	MashZone NextGen authentication is configured for basic credentials.
presto_username	The user name for basic credentials for this request. The value must be passed in clear text. Typically, this header should only be used, in combination with the `presto_pawwsord` header, to log in or authenticate a new principal when MashZone NextGen authentication is configured for basic credentials.
serviceHeader	**Note:** Only applicable to requests using the JUMP protocol. An object to forward as a header in a JUMP request to a mashable information source. A common example would be a SOAP header for a WSDL web service.
x-p-anonymous	Indicates that this request can be treated as a guest user, with no credentials.
x-p-dom	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-dow	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-enableSnapshot	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-httpBasicAuth	An object with the user credentials (username and password properties) to send to a mashable in an HTTP basic authentication header. If the mashable was registered in MashZone NextGen with a basic authentication security profile, these credentials override the default credentials.
x-p-hour	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-method	`delete` or `put` The MashZone NextGen Server does not support HTTP DELETE or PUT requests. This header

	indicates that this request is to delete or a save the subject of the request. **Note:** This header is *only* valid if the specific mashable supports HTTP DELETE or PUT or the MashZone NextGen API supports this header.
x-p-minute	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-ntlmAuth	An object with the user credentials (username, password and Windows domain properties) to send to a mashable. If the mashable was registered in MashZone NextGen with an NTLM security profile, these credentials override the default credentials.
x-p-paginateCount	The total number of items to include in one page of results for a paginated result set. See also `x-p-paginateStart`.
x-p-paginateRecord	An XPath expression identify the repeating node in results that defines one "item" to return in paginated result sets. **Note:** If there are no matches to this path, the full response from the mashup or mashable is returned.
x-p-paginateStart	An index number for the first item in a paginated result set to return in the response to this request. If this is 1, the request invokes the mashable or mashup and the MashZone NextGen Server caches the full result. The first *N* items are returned up to the value of `x-p-paginateCount`. If this value is greater than 1, the response returns the set of items requested from cache.
x-paginateToken	The pagination token returned by the MashZone NextGen Server that is used to coordinate multiple requests for paginated results from mashables or mashups. This is a response header returned when the mashable or mashup is first invoked. All subsequent requests for additional pages of results should include this header.

x-p-portType	The port type name to use for WSDL web services whose WSDL has multiple port types.
x-p-renameSOAPPrefixes	`true` | `false` A flag to have the MashZone NextGen Server change the namespace prefix used for the root element in SOAP responses to `presto`. All descendant elements with that namespace are also affected. This flag is false, by default. Use it to normalize the prefix for WSDL services that use dynamic namespace prefixes for each request.
x-p-repeatcount	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-repeatinterval	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-resultContentType	The MIME type or Internet Media type, such as `text/plain; charset=UTF-8`, to use for the result. By default, the character encoding for text responses is UTF-8. For text media types, you can set the character encoding you want to receive as the result. You could change this to `text/plain; charset=ISO-8859-1`, for example, to get the result in a Latin-1 character encoding.
x-p-resultFeedType	Sets the syndicated web feed type and version that the response should be returned in: ■ `RSS_0.9` ■ `RSS_0.91` ■ `RSS_0.92` ■ `RSS_0.93` ■ `RSS_0.94` ■ `RSS_1.0` ■ `RSS_2.0` ■ `Atom_0.3` ■ `Atom_1.0`

	This header overrides the default normalization setting for the MashZone NextGen Server set by MashZone NextGen administrators. See "Syndicated Feeds (RSS/Atom)" on page 131 for the specific limitations and impacts of normalization.
x-p-resultFormat	Defines the format for the mashable or mashup response: ■ `csv` = comma-separated values. ■ `json` = JSON (JavaScript Object Notation). ■ `native` = the default format returns a response in the form native to the information source. ■ `xml` = XML.
x-p-returnHeaderOnly	`true` | `false` A flag for *mashups only* to suppress invoking any mashables and just return HTTP headers, such as the `serviceURL` with the REST URL to invoke this mashable information source or mashup.
x-p-returnServiceHeaders	`true` | `false` A flag to control whether HTTP headers in the response from a mashable are included in the response from MashZone NextGen. This header also affects the `serviceURL` response header which contains the REST URL that corresponds to this request. Out of the box, this flag defaults to `true`, however MashZone NextGen administrators can disable this feature. This flag is only needed to override the default behavior defined in Mashup Server configuration. If the server does not forward mashable headers by default, setting this flag in a request will ensure that HTTP headers are sent and vice versa. The headers that can be forwarded are defined in a white list in MashZone NextGen Server configuration. To ensure that a specific header is forwarded, a MashZone NextGen administrator may need to update MashZone NextGen Server configuration.

x-p-returnServiceURLWithCredentials	`true` | `false` A flag. If true, this overrides the default behavior and returns actual user credentials in the `serviceURL` response header. The `serviceURL` response header contains the REST URL for this mashable or mashup. By default, the user credentials in this URL have references to session parameters rather than actual user credentials.
x-p-scheduler	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-serviceEndpoint	For WSDL web services *only*, this is the URL that should override the existing endpoint for the web service in this request. Use this header to handle dynamic endpoints for WSDL services.
x-p-sessionbinding	An object with one or more tokens that define portions of the response to this request that should be saved in the session. Later requests can use these tokens as input parameters to be resolved from session data. Session binding tokens for this header are in the form: `@sessionbinding.`*`attribute-name=XPath/to/nodes/to/save`
x-p-sharepoint*	MashZone NextGen headers beginning with this prefix are reserved for internal use with MashZone NextGen Add-On for SharePoint.
x-p-snapshotName	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.

MashZone NextGen Response Headers

MashZone NextGen returns some information in response headers, typically when requested by headers in the request.

serviceURL	The REST URL that corresponds to the request that generated this response. This URL invokes the same mashable or mashup, with any relevant input parameters or MashZone NextGen headers. By default, this URL does not include user credentials

	unless specifically requested with an optional MashZone NextGen header in the request.
serviceHeader	**Note:** Only applicable to responses using JUMP. An object containing headers returned by the mashable, such as SOAP headers.
snapshotToken	The ID of the snapshot that was created from this request. See "Take or Delete Snapshot Methods" on page 1518 for more information.
x-paginateToken	The pagination token that identifies the result set for this response. The MashZone NextGen Server uses this token to identify which cached results to use when additional pages of results are requested. All subsequent requests should include this token as a header or parameter.
x-presto-cache	Indicates that this response has been supplied from the MashZone NextGen Response Cache.
x-totalCount	The total number of items in the cached result set for this pagination token.

Obsolete JUMP Headers

The following optional JUMP headers from versions 2.7 and earlier are obsolete and no longer supported in versions 3.0 and above:

authToken	An authentication token for requests from clients that have already logged in. Authentication has changed in version 3.0 and above to use a standard HTTP session.
password	See presto_password.
username	See presto_username.

MashZone NextGen Snapshot API

Users can take snapshots from artifact pages or schedule jobs to take snapshots automatically. These snapshots can be registered as mashable information sources and

used individually to create basic apps. (See "Snapshot Mashables" on page 130 for links to topics on user functions that create or use snapshots.)

In many cases, however, snapshots are meant to be used as a set, showing trends or other statistics over a period of time or over other variables. To include this information in a mashup or custom app, you use the Snapshot API to retrieve snapshots.

This is a REST API that builds on the MashZone NextGen REST API. You can use these API methods:

- Take or Delete Snapshot Methods
- Get Snapshot Methods
- Schedule Snapshot Job Methods and Manage Scheduled Snapshot Job Methods

Take or Delete Snapshot Methods

There are two different methods that you can use to take snapshots programmatically:

- Take a Snapshot Immediately
- Take a Temporary Snapshot and Save Separately

You can also Delete Snapshots.

Take a Snapshot Immediately

You can take an immediate snapshot for any mashable or mashup using the MashZone NextGen REST URL to run the mashable or mashup plus two MashZone NextGen headers: `x-p-enableSnapshot` and `x-p-snapshotName`.

For example:

`http://`*app-server:port* `/presto/edge/api/rest/`*service* */operation* `?`*service-params-if-any* `&x-p-enableSnapshot=true&x-p-snapshotName=`*name-or-uuid*

You can find the REST API URL for any mashables or mashup in the Technical Specs tab on its artifact page. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information. Add any input parameters needed to run this mashable or mashup or any other "MashZone NextGen Headers/Parameters" on page 1511, as needed.

Parameters

Parameter	Req?	Description
`x-p-enableSnapshot`	yes	`true` = determines that the response for the mashable or mashup should create a snapshot.
`x-p-snapshotName`	yes	Choose either: ■ A specific name for this snapshot.

Parameter	Req?	Description
		■ `uuid`, to have MashZone NextGen automatically assign a unique ID to this snapshot.

Take a Temporary Snapshot and Save Separately

With this pattern, you create a temporary snapshot when you run a mashable or mashup and choose to save it permanently in a separate request.

Note: Temporary snapshots must be saved within 25 minutes of their capture. After that, the MashZone NextGen Server discards the temporary snapshot.

You use the MashZone NextGen REST URL to run the mashable or mashup, but include only the `x-p-enableSnapshot`MashZone NextGen header. For example:

`http://`*app-server:port* `/presto/edge/api/rest/`*service* */operation* `?`*service-params-if-any* `&x-p-enableSnapshot=true`

Add input parameters to the URL or a request body, as needed, plus any other MashZone NextGen headers you need. You can find the REST API URL for any mashables or mashup in the Technical Specs tab on its artifact page. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information.

MashZone NextGen assigns a snapshot ID to the temporary snapshot and returns this ID in the `snapshotToken` HTTP header of the response, such as this example:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
...
cache-control: private, max-age=1200
serviceURL: http://localhost:8080/presto/edge/api/rest/YahooWeatherREST/getData?p%3D94572
snapshotToken: 7ac5fe05-2e67-11e0-952d-adff03bd533b
...
```

When you are sure you want to save this as a permanent snapshot, send an HTTP POST request to this URL:

`http://`*app-server:port* `/presto/edge/api/rest/snapshot`

Use the following POST body with the snapshot ID returned when the snapshot was taken and the appropriate metadata for this snapshot.

POST Body

```
<snapshot xmlns="http://www.jackbe.com/2009-08-01/snapshot"
  id="a2da57e1-2a60-11e0-9f1b-69cef3c615e5">
  <name>mySnapshot</name>
  <service>mashableA</service>
  <operation>operationName</operation>
  <createdby>userA</createdby>
  <description>Snapshot of some kind of data...</description>
  <tags>
    <tag>tag1</tag>
    <tag>tag2</tag>
    <tag>tag3</tag>
```

```
  </tags>
</snapshot>
```

Delete Snapshots

To delete a snapshot, use a URL in this form:

`http://`*app-server:port* `/snapshot/`*snapshot-id* `?x-presto-method=DELETE`

This uses the MashZone NextGen header/parameter `x-p-method`. See "Get Snapshot Methods" on page 1520 for more information on obtaining snapshot IDs.

Get Snapshot Methods

There are two query methods you can use to retrieve snapshots:

- Get Snapshot Collections By Criteria
- Get One Snapshot By ID

Typically, you must query for snapshots based on criteria to get the ID(s) for the snapshot(s) you want to retrieve. Then retrieve each snapshot based on its ID.

Get Snapshot Collections By Criteria

Base URL

To retrieve all snapshots, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/snapshots`

In most cases, you should add Parameters to filter the response to a specific set of snapshots. You can also use these parameters to sort the snapshot references in the response See Snapshot Collection Query Examples for some common combinations.

The response when you find a collection of snapshots contains a link with the URL to retrieve each matching snapshot. See Snapshot Collection Responses for an example.

Parameters

You can add any combination of the following parameters to queries for snapshot collections except that you must specify `service` if you specify `operation`.

Parameter	Req?	Description
`name`		A full or partial snapshot name. Snapshots whose name begins with this value are included.
`tag`		A tag to match.
`service`		The name of the mashable or mashup that this snapshot was generated from. You can use this

Parameter	Req?	Description
		parameter by itself or in conjunction with the `operation` parameter. The name is case sensitive. You can also use partial artifact names. This matches artifacts whose name begins with the value you specify.
`operation`		The operation for the mashable or mashup that this snapshot was generated from. You must use this parameter in conjunction with the `service` parameter.
`createdby`		The name of the user who generated this snapshot.
`fromdate`		The first scheduled execution date to include in the form: `YYYY-MM-DD` See Snapshot Collection Query Examples for an example.
`todate`		The first scheduled dateto exclude in the form: `YYYY-MM-DD` See Snapshot Collection Query Examples for an example.
`sortby`		The field to use to sort snapshot references by in the response. You can sort by: - `name` - `tags` - `service` - `operation` - `createdby` (default)
`sortdir`		The sort order to use, if you use the `sortby` parameters. This can be `ascending` (default) or `descending`.

Snapshot Collection Query Examples

Get snapshots by name or partial name:

```
http://localhost:8080/presto/edge/api/rest/snapshots?name=mySnap
http://localhost:8080/presto/edge/api/rest/snapshots?name=my
```

Get all snapshots for all operations of one service:

```
http://localhost:8080/presto/edge/api/rest/snapshots?service=YahooWeatherREST
```

Get all snapshots for one operation of a service:

```
http://localhost:8080/presto/edge/api/rest/snapshots?service=XigniteHistorical&operation=GetTopMo
```

Get all snapshots created by one user:

```
http://localhost:8080/presto/edge/api/rest/snapshots?createdby=JAgusi
```

Get all snapshots for a tag and sort by artifact name:

```
http://localhost:8080/presto/edge/api/rest/snapshots?tag=news&sortby=service
```

Get all snapshots created since a specific start date:

```
http://localhost:8080/presto/edge/api/rest/snapshots?fromdate=2011-01-31
```

Get all snapshots created on one specific date:

```
http://localhost:8080/presto/edge/api/rest/snapshots?fromdate=2011-01-31&todate=2011-02-01
```

Get all snapshots created by one user using a specific tag

```
http://localhost:8080/presto/edge/api/rest/snapshots?createdby=ladams&tag=projectA
```

Snapshot Collection Responses

Snapshot collections contain metadata for matching snapshots but no snapshot data. This metadata includes the snapshot ID as well as the full URL (in `content/@href`) that you can use to retrieve individual snapshots.

```
<snapshots xmlns="http://www.jacbe.com/2009-08-01/snapshot">
  <snapshot id="9445ae9b-28d3-11e0-9c7b-f391b2692a54">
    <name>Snapshot - Tue Jan 25 2011</name>
    <service>XigniteHistorical</service>
    <operation>GetTopMovers</operation>
    <createdby>Jane</createdby>
    <description>10 top movers 12/31/2010</description>
    <tags><tag>movers</tag></tags>
    <view>Grid</view>
    <createdtime>2011-01-25T14:37:41.935-08:00</createdtime>
    <createdtimemillis>1295995061960</createdtimemillis>
    <annotations count="0"/>
    <content type="uri"
href="/presto/edge/api/rest/snapshot?id=9445ae9b-28d3-11e0-9c7b-f391b2692a54"
/>
  </snapshot>
  <snapshot id="7f6cb381-298e-11e0-afe3-8dcf3e37fbe1">
    ...
  </snapshot>
  ...
</snapshots>
```

Get One Snapshot By ID

Base URL

To retrieve a specific, complete snapshot, you must have the snapshot ID. Typically, you find this ID in the response to a snapshot collection query. See "Get Snapshot Collections By Criteria" on page 1520 for an example of this response.

You use the URL in `content/@href` from the snapshot collection query response to retrieve the snapshot you want. This URL is in the form:

`http://`*app-server:port* `/presto/edge/api/rest/snapshot/`*snapshot-id*

A Single Snapshot Response

This example shows a snapshot in XML format. The actual results in the snapshot are contained in the `content` element:

```
<snapshot xmlns="http://www.jackbe.com/2009-08-01/snapshot"
    id="747523af-2e50-11e0-bde1-61d07a0308ed">
  <name>Top Losers Dec 31, 2010</name>
  <service>XigniteHistorical</service>
  <operation>GetTopLosers</operation>
  <createdby>admin</createdby>
  <description>Top stock lowers for 12/31/2010</description>
  <tags>
    <tag>losers</tag>
  </tags>
  <view>Grid</view>
  <createdtime>2011-02-01T14:16:07.928-08:00</createdtime>
  <createdtimemillis>1296598567932</createdtimemillis>
  <content xmlns="">
     <!-- actual snapshot data -->
    <GetTopLosersResponse xmlns="http://www.xignite.com/services/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
     <GetTopLosersResult>
     ...
    </GetTopLosersResponse>
  </content>
  <annotations/>
  <filter/>
</snapshot>
```

Schedule Snapshot Job Methods

Base URL

You can schedule a job to take snapshots at regular intervals for any mashable or mashup. Use the MashZone NextGen REST URL to run the specific mashable or mashup as the base URL to schedule a job. You can find this URL in the Technical Specs tab on the artifact page for any mashables or mashup.

Include input parameters, if any, for the mashable or mashup in the URL and any other MashZone NextGen headers, as needed.

Then add the `x-p-enableSnapshot` and `x-p-snapshotName` parameters. For example:

`http://`*app-server:port* `/presto/edge/api/rest/`*service* */operation* `?`*service-params-if-any* `&x-p-enableSnapshot=true&x-p-snapshotName=`*name-or-uuid*

To this base URL you add appropriate Parameters that define the schedule for the jobs.

Important: It is a best practice to limit snapshot schedules to a specific number of repetitions, although this is not required. Open-ended schedules can have a negative impact on performance.

See Schedule Examples for some combinations of these parameters.

Parameters

You must specify the schedule for the job and any additional parameters required for that type of schedule.

Parameter	Req?	Description
`x-p-enableSnapshot`	yes	Determines that this invocation is to create a scheduled job.
`x-p-snapshotName`	yes	Choose either: ■ A single name to assign to all snapshots taken by this job. ■ `uuid`, to have MashZone NextGen automatically assign a unique ID to each snapshot for this job.
`x-p-scheduler`	yes	The type of schedule for this job. Valid schedule types include: ■ `secondly` = schedule every second. Or schedule every number of seconds defined in `x-p-repeatinterval` with a schedule termination in `x-p-repeatcount`. ■ `minutely` = schedule every minute. Or schedule every number of minutes defined in `x-p-repeatinterval` with a schedule termination in `x-p-repeatcount`. ■ `hourly` = schedule every hour. Or schedule every number of hours defined in `x-p-repeatinterval` with a schedule termination in `x-p-repeatcount`. ■ `daily` = schedule every day at a specific time. You must include the `x-p-hour` and `x-p-minute` parameters with a schedule of this type.

Parameter	Req?	Description
		■ `weekly` = once a week, on a specific day of the week at a specific time. You must include the `x-p-dow`, `x-p-hour` and `x-p-minute` parameters with a schedule of this type. ■ `monthly` = once a month, on a specific day at a specific time. You must include the `x-p-dom`, `x-p-hour` and `x-p-minute` parameters with a schedule of this type.
`x-p-hour`		The hour, from 0 to 23, to run a daily, weekly or monthly scheduled job. You must also add the `x-p-minute` parameter.
`x-p-minute`		The minute to run a daily, weekly or monthly scheduled job. You must also add the `x-p-hour` parameter.
`x-p-dow`		The day of the week, from 1-7 beginning on Sunday, to run a weekly scheduled job. You must also add the `x-p-hour` and `x-p-minute` parameters.
`x-p-dom`		The day of the month (1-31) to run a monthly scheduled job. You must also add the `x-p-hour` and `x-p-minute` parameters.
`x-p-repeatcount`		The maximum number of scheduled snapshots to take for this scheduled job. You must also use the `x-p-repeatinterval` parameter. **Tip:** Is it highly recommended to include this optional parameter in job schedules that use second, minute or hourly intervals to define a termination point for the job.
`x-p-repeatinterval`		The number of time measurements between each scheduled snapshot for a scheduled job. The actual time measurement is defined by the value of the `x-p-scheduler` parameter, such as `hourly`. You must also specify the `x-p-repeatcount` parameter to define a limit for the schedule.

Schedule Examples

These examples all use the sample mashable Yahoo Weather Given Zipcode which is a REST mashable that takes a single input parameter, `p`, with a zip code:

- Schedule jobs every 3 hours for a 24 hour period:

  `http://localhost:8080/presto/edge/api/rest/YahooWeatherREST/getData?p=98106&x-p-enableSnapshot`

- Schedule a job on Wednesday each week at 6:00p.m., with no termination:

  `http://localhost:8080/presto/edge/api/rest/YahooWeatherREST/getData?p=98106&x-p-enableSnapshot`

Manage Scheduled Snapshot Job Methods

You can retrieve information about scheduled snapshot jobs (see "Find Jobs" on page 1526) or manage jobs (see "Suspend Jobs" on page 1526, "Resume Jobs" on page 1527 and "Delete Jobs" on page 1527).

Find Jobs

To retrieve information about scheduled snapshot jobs, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/jobs?`

Add the following parameters as needed to define the scope for this query.

Parameter	Req?	Description
`user`	yes	The name of the user whose job schedules should be returned.
`service`	yes	The name of the mashable or mashup to retrieve job information for.
`operation`		An optional name of the specific operation for this mashable or mashup to retrieve job information for.

This example retrieves jobs for a specific mashable operation and user:

```
http://localhost:8080/presto/edge/api/rest/meta/jobs?user=sdiggle
&service=employees-table&operation=findByID
```

This example retrieves jobs for all operations of a specific mashable and user:

```
http://localhost:8080/presto/edge/api/rest/meta/jobs?user=admin
&service=XigniteHistorical
```

Suspend Jobs

To suspend the scheduled execution of all snapshot jobs, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/suspendall`

To suspend the schedule execution of a specific snapshot job, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/suspendjob?user=`*username* `&job=`*jobname*

Both the `user` and `job` parameters are required.

Resume Jobs

To resume the scheduled execution of all snapshot jobs, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/resumeall`

To resume the schedule execution of a specific snapshot job, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/resumejob?user=`*username* `&job=`*jobname*

Both the `user` and `job` parameters are required.

Delete Jobs

To delete scheduled executions of a snapshot job, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/job/`*username* */jobname* `?x-presto-method=DELETE`

This uses the MashZone NextGen header/parameter `x-p-method`.

About DataTable Events

Mashable and mashup results used with views in basic apps are held in a *DataTable*, a client-side data model. See the MashZone NextGen DataTable API for details.

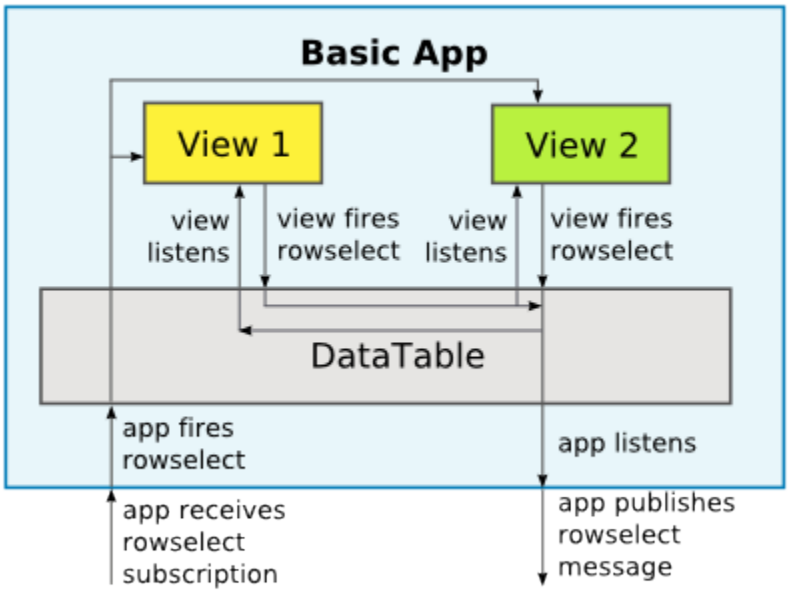
Each basic app has a DataTable with results for their mashable or mashup. This includes basic apps created in App Maker and views that have been directly added to workspace apps if they are configured to share data (see "Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace" on page 1125).

Built-in and pluggable views included in a basic app interact with result data through the app's DataTable. Custom apps can also use DataTables, if desired.

In addition to basic data access, DataTables support result pagination, track selection state in result rows and trigger events for row selection. DataTables use a single-row selection model.

DataTable Events for Views and Apps

DataTables have a single event, `rowselect`, of interest to views and basic apps. This event indicates that one row of data in results has been selected by the user, typically through the user interface of a view within the app. If the app is part of a workspace, however, this change in row selection can also come from messages published by another app that are wired to the `rowselect` subscription topic.



Views can both trigger and listen for DataTable `rowselect` events, allowing all interested views that are using a DataTable to synchronize their selection state across the app. Basic apps also listen for `rowselect` events in the DataTable and then publish messages to the `rowselect` topic or trigger `rowselect` events in the DataTable as the result of messages received from other apps in the `rowselect` subscription topic. Subscriptions to `rowselect` topics can also be used to synchronize app state when the workspace initially loads (see "Synchronizing Wired Apps When a Workspace Loads" on page 1181 for information).

Note: Wiring in a workspace app that ties `rowselect` events in the DataTable for one app to row selection for DataTables in other apps is possible, but may not always be consistent. Pagination in a DataTable can cause matches to be missed. Multiple matches will also not be found as the selection model for the DataTable is a single selection.

MashZone NextGen built-in views that support some type of 'click' event both trigger and listen for DataTable `rowselect` events. Pluggable views can also choose to trigger or listen for this event. See "Pluggable View Classes: Triggering and Handling Events" on page 1059 for more information.

A `rowselect` event includes these properties:

- `rowIndex` = the index number of the row for this event
- `selected` = a boolean indicating whether this row should be selected (true) or not (false)
- `rowData` = a row object with the data for this row

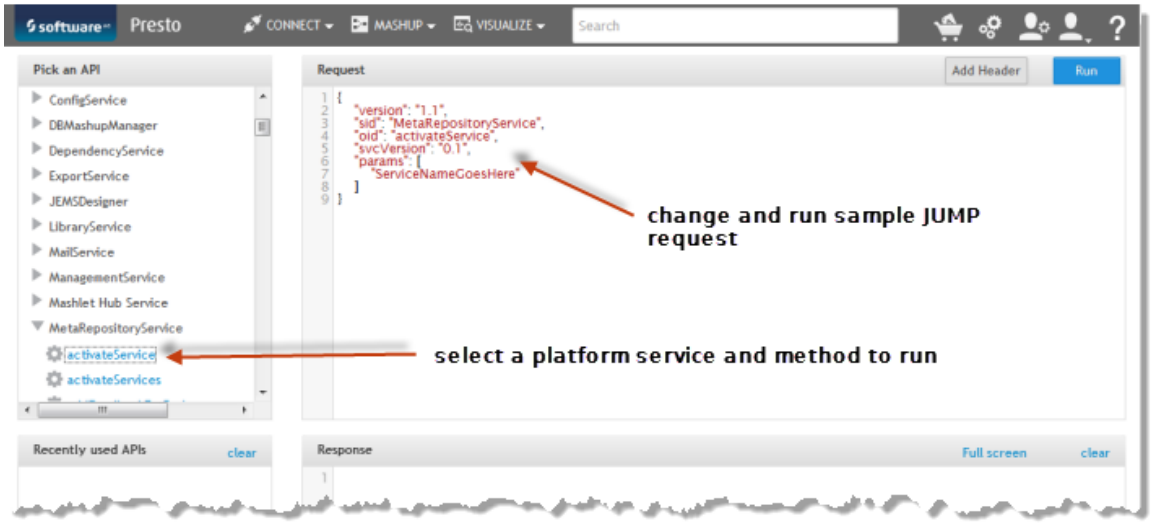
When a view indicates a row has been clicked, the DataTable fires one or two `rowselect` events:

- For the existing selected row, if any, an event with `selected = false` to clear selection for this row.
- If the clicked row is not the existing selected row, an event with `selected = true` to set the selected state for this new row.

MashZone NextGen Platform API Console

This console allows MashZone NextGen administrators and developers to discover and run platform APIs for MashZone NextGen. Platform APIs can be useful to investigate issues that you may report to Technical Support. They may also be used in some cases during upgrades to a new version of MashZone NextGen. Platform APIs can also be used to provide specialized features or integration.

Click in the main menu to open the API Console:



Platform APIs are organized in services that focus on specific aspects of MashZone NextGen. Find and select the MashZone NextGen Platform service and method you want to run. In most cases, the service names are self-explanatory. See "MashZone NextGen Platform Services" on page 1530 for more information if needed.

Once you select a platform API method, a sample request to invoke this method fills the Request pane.

Note: Most platform APIs use the JUMP (JackBe Universal Message Protocol) format. See "The JackBe Universal Message Protocol (JUMP)" on page 1531 for more information on this format.

Click **Run** to send the request. The JUMP response from the MashZone NextGen Server displays in the Response pane. See "JUMP Responses" on page 1532 for more information on responses.

MashZone NextGen Platform Services

Service	Description
AppSearch	Contains methods to find apps or workspace apps based on a variety of characteristics.
AppService	Contains methods to manage apps and workspace apps including add, remove, activate, publish and so on. This service also provides methods to get app specifications or specific portions/properties of app specifications.
AppStoreService	Contains methods to manage apps in the AppDepot including submit, approve, reject, add to favorites and so on. Also contains methods to find apps published to AppDepot based on various characteristics.
CollabService	Contains methods around collaboration in MashZone NextGen Hub including activity, commenting, broadcast messages, rating, usage and so on.
ConfigService	Contains methods to get and set configuration properties for the MashZone NextGen Server. Includes both methods for specific properties and generic methods to get and set properties by name.
DBMashupManager	Contains methods to manage drivers, datasources and the database services based on registered datasources.
DependencyService	Contains methods to track and manage dependencies between MashZone NextGen artifacts.
ExportService	
JEMSDesigner	Contains methods to manage macros and macro domains.
LibraryService	Contains methods to find and manage libraries and other resources used in pluggable views.
MailService	Contains methods used to manage and send email notifications for sharing.

Service	Description
ManagementService	Contains methods to find license and statistics information for the Admin Console.
MetaRepositoryService	Contains methods to manage mashables, mashups, categories, providers and MashZone NextGen global attributes including add, register, activate, find, validate and so on.
PolicyService	Contains methods to find and manage permission sets for run permissions, other artifact actions and access to specific features (entitlements) in MashZone NextGen Hub and the AppDepot. Also contains methods to create and manage dynamic groups for custom policies and manage policy configuration.
ProfileService	Contains methods to find and manage user profiles for Mashboard. Mashboard profiles save the state of each user's currently open workspaces.
ResourceService	Contains methods to find and manage file resources used in apps, pluggable libraries, custom Wires blocks or other extensions to MashZone NextGen.
Sample Requests with Test Data	A set of sample requests to invoke various mashables.
ShareManager	No longer used.
SharePointManager	Contains methods to get connections, sites, lists and other configuration used to register Lists or use SharePoint as a source of information in mashables or mashups.
UserManagerService	Contains methods to find and manage users, groups and MashZone NextGen users attributes.

The JackBe Universal Message Protocol (JUMP)

JUMP is a deprecated protocol used in previous MashZone NextGen releases for requests to and responses from MashZone NextGen to invoke mashables or mashups. However, JUMP is still used extensively in the MashZone NextGen platform APIs.

JUMP is based on the JSON (JavaScript Object Notation) format to provide a compact wire presence. JSON represents objects as a hierarchy of name/value pairs using braces ({ }) and other simple delimiters to define the syntax. Values can be simply types, objects or arrays. See the json.org site for a good introduction to the JSON syntax.

JUMP Requests

JUMP requests contain a set of properties for:

- *General information* for invocation of a specific mashable, mashup or platform API and optional headers.

  The optional headers request specific behavior, such as receiving responses in an XML format or a header that should be passed to the mashable to be invoked. For a complete list of optional JUMP headers for requests, see "MashZone NextGen Headers/Parameters" on page 1511. For examples using optional JUMP headers, see "Adding Headers for Mashables to JUMP Requests" on page 1535.

- *Input parameters* for the platform API, mashable or mashup. The `params` property may be an array of zero or more ordered parameters or an object with named parameters in JSON format that should be passed to the API, mashable or mashup that is being invoked. Parameters may be simple types, arrays or objects. See "Input Parameters for JUMP Requests" on page 1534 for more information.

For example:

```
{
  "version":"1.0",
  "sid":"RDSService",
  "svcVersion":"0.1",
  "oid":"getAllServiceTypes",
  "params":[]
}
```

JUMP Responses

Responses contain a set of properties whose values are in JSON format, with the possible exception of the `response` property. This includes:

- *General information* about the request for this response, such as the serviced ID (`sid`), operation ID (`oid`), JUMP version (`version`), `svcVersion` (reserved for future use) and invocation ID (`invId`). For example:

  ```
  { "sid":"RDSService",
     "oid":"getAllServiceTypes",
     "svcVersion":"0.1",
     "invId":"",
     "version":"1.0",
    ...}
  ```

- *User, session and optional headers* in the `header` property. For example:

  ```
  { "sid":"RDSService",
    ...
    "header":{
      "map":{
        "currentUser":"admin",
  ```

```
      "sessionTimeout":"1800",
      "serviceHeader":{
        "map":{}
      }
    }
  },
  ...}
```

The `header` property is an object with a `map` property which is also an object that contains `currentUser`, `sessionTimeout` and optional properties.

If the request was for an authenticated user, `currentUser` contains the username for this session. If the request was anonymous, or the current session for the user has timed out, `currentUser` is `anonymousUser`.

The `sessionTimeout` property contains the number of seconds until this session expires. Subsequent requests for authenticated or anonymous users extend the session.

Optional JUMP response headers may include headers returned from a mashable, pagination tokens or other information requested from the MashZone NextGen Server. See "MashZone NextGen Headers/Parameters" on page 1511 for a complete list.

- *Response data* in the `response` property if the invocation was successful. The mashable or API response may be in JSON format (by default), in XML or in a native format for this mashable. An optional JUMP header in the request can choose another format. For example:

```
{ "sid":"RDSService",
  ...
  "error":"",
  "errorCode":"",
  "response":[ { "type":"EXCEL", "label":"EXCEL" },
    { "type":"REST", "label":"REST" },
    { "type":"WSDL", "label":"WSDL" },
    { "type":"MASHUP", "label":"MASHUP" },
    { "type":"RSS\/Atom", "label":"RSS\/Atom" },
    { "type":"DAO", "label":"DAO" } ]
}
```

- *Error information* if the request failed or the service invocation failed. This includes the error message (`error`) from the platform API, mashable, mashup or from MashZone NextGen and an error code (`errorCode`) from MashZone NextGen, if the request failed in MashZone NextGen. These properties are null or empty if the `response` property has content.

  For example:

```
{
  "sid":"HelloWorldService",
  "appId":"",
  "oid":"sayHello1",
  "svcVersion":"0.1",
  "error":"Failed to handle given request. Error = Unable to find service using (HelloWorldSe
  "errorCode":"",
  "invId":"",
  "response":"",
  "version":"1.0"
}
```

Input Parameters for JUMP Requests

JUMP requests contain a `params` property that takes either an array of parameters or an object with named parameters to pass to mashables, mashups or to platform service APIs. There are several ways to construct this array to accommodate different requirements:

- *No parameters* = an empty array, such as:

  ```
  params: [ ]
  ```

- *Unnamed, ordered parameters with simple types* = array entries are separate by commas. They can be any legal JSON value or a mapping expression identifying aMashZone NextGenattribute, such as:

  ```
  params: [ "short", 10 ]
  ```

  or

  ```
  params: [ "$global/AmazonId", true ]
  ```

  See "Parameter Values" on page 1535 for details.

- *Named parameters with simple types in any order* = an object with matching properties for each named parameter, such as:

  ```
  params:  { "itemsPerPage": 10,
      "username": "$user/rssUserID" }
  ```

  The values for individual parameters (each property value) can be any legal JSON value or a mapping expression identifying a MashZone NextGen attribute. See "Parameter Values" on page 1535 for details.

  Parameter names can also use namespaces, where required, as shown in the (following) example for a complex parameter.

- *A single, unnamed, complex parameter with named properties* = an object with any level of complexity and named properties.

  This is a special case of an unnamed parameter that is represented by a complex object with named properties. This example contains a complex object where properties may also have objects as values. Property names also include a namespace prefix:

  ```
  params: { "ns:ItemLookup":
             { "ns:SubscriptionId":"0525E2PQ81DD7ZTWTK82",
               "ns:Validate":"False",
               "ns:Request":
                 { "ns:ItemId":"B000EXS1BS",
                   "ns:IdType":"ASIN",
                   "ns:ResponseGroup":"Large"
                 }
             }
           }
  ```

Parameter Values

Explicit parameter values, either named or unnamed, can be any type valid in JSON:

- *String* = characters enclosed in double quote marks, such as "string".
- *Number* = numeric characters with no quote marks, such as 104.2.
- *Keywords* = `true`, `false` or `null`
- *Arrays* = enclosed in brackets with commas separating array entries, such as [10, 5, 0].
- *Objects* = enclosed in braces with named properties. Property names are strings and end in a colon followed by the property value. For example, { propertyA: true } or { "ns:searchType": "book" }.

Note: Property names that include XML namespaces must be enclosed in double quote marks to ensure that the colon for the namespace is clear.

See the http://www.json.org/json.org site for a good introduction to JSON types and syntax.

Parameter values can also be resolved by the MashZone NextGen Server from a MashZone NextGen attribute. The value passed in the request is a mapping expression that identifies the MashZone NextGen attribute to use to resolve this input parameter. See "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502 for more information.

Adding Headers for Mashables to JUMP Requests

Some mashables require headers in requests. For example, WSDL web services may require some information in SOAP headers that are separate from the body of the request.

You can add SOAP headers or other headers required by a mashable to JUMP requests using the optional JUMP header property *serviceHeader*.

SOAP Header Example

The following example is a JUMP request for a WSDL web service that passes authentication information in a SOAP header:

```
{ "sid":"XIgniteGetQuotes",
  "oid":"GetQuickQuotes",
  "header":{
    "serviceHeader":{ // jump parameter to pass SOAP header
      "ser:Header":{ // one SOAP header to pass to SOAP service
        "ser:Username":"noname@somewhere.com"
      }
    }
  },
  "params":{ "ser:GetQuickQuotes": {"ser:Symbol":"MSFT"} }
  }
```

Namespaces and Namespace Prefixes

JUMP is based on the JSON protocol which does not understand or directly support XML namespaces. Namespaces are used heavily in SOAP and SOAP headers.

To correctly identify the elements of a SOAP header, you *must*:

- Use the XML namespace *prefix*that is defined in the schema for that element as the prefix for the property name in the `serviceHeader`object.

  In the previous example, `ser` is the namespace prefix defined for SOAP headers in the schema for this service.

- Enclose the fully-qualified name (the namespace, colon and element name) in quotation marks for each SOAP header element to ensure that the colon in the qualified name is not misinterpreted as part of the JSON syntax.

  For example, `"ser:Username"` is a property name but `ser:Username` is a property (ser) and value (Username) in JSON syntax.

MashZone NextGen APIs

MashZone NextGen has several APIs that you can use to connect to MashZone NextGen, to create pluggable views or custom apps or to extend MashZone NextGen features. Reference documentation for these APIs is available from the following links.

MashZone NextGen REST APIs and MashZone NextGen Connect

Applications can use the following APIs to connect to the MashZone NextGen Server and invoke MashZone NextGen mashables or mashups:

- "MashZone NextGen REST API" on page 1509
- MashZone NextGen Connect for JavaScript
- "MashZone NextGen Snapshot API" on page 1517

See also "MashZone NextGen Headers/Parameters" on page 1511 that can be used with these APIs.

Pluggable Views, Custom Apps and Common JavaScript Client Libraries

When the MashZone NextGen built-in views do not provide the visualizations you need, you can add pluggable views to the MashZone NextGen View Gallery and the MashZone NextGenView Maker using the MashZone NextGen Views API.

Pluggable views use a *datatable* from the MashZone NextGen DataTable APIMashZone NextGen DataTable API as the interface to mashable or mashup results. This API

provides a light-weight, tabular client-side data model that you can also use in custom apps to work with mashable or mashup results.

When the basic apps generated by App Maker do not provide needed functionality, you can use the MashZone NextGen App API in conjunction with the App Specifications to define custom apps.

The MashZone NextGen Library Loader API provides methods to easily load JavaScript or CSS libraries automatically and seamlessly handle any dependencies. Both the MashZone NextGen View Framework and the MashZone NextGen Apps Framework use the Library Loader to load dependencies for view or apps, including third-party libraries that are bundled in MashZone NextGen and any pluggable libraries that you add to MashZone NextGen. Typically, pluggable libraries include code from your organization or other third parties that you provide to extend base functionality, such as libraries for pluggable views.

Miscellaneous Extension APIs

Extension APIs for MashZone NextGen provide extension points to accommodate application requirements. These include:

- Custom XPath Functions for Mashup Scripts.

  This API allows you to define custom XPath functions to perform calculations or transformations within XPath expressions in mashup scripts.

- User-Defined Analytic Functions for large dataset queries with RAQL.

  This API allows you to define window analytic function or aggregate analytic functions in addition to the built-in MashZone NextGen functions that you can use in RAQL queries in mashup scripts. You can also define plain functions for RAQL queries which do not require this API.

- Custom SSO Filters for MashZone NextGen Authentication with Single Sign-On solutsions.

  This API allows you to extend the default identity extraction behavior of MashZone NextGen when authentication is integrated with a Single Sign-On solution.

- Custom Certificate Validation for MashZone NextGen Authentication with Digital Certificates.

  This API allows you to extend the default certification validation behavior of MashZone NextGen when authentication uses digital certificates.

- Custom Security Profiles for mashable information sources.

  MashZone NextGen provides security profiles for the most common types of secure connections including basic authentication, NTLM, SSL or single sign-on. This API allows you to define and handle custom authentication or secure connection requirements with mashable information sources.

MashZone NextGen REST API

The MashZone NextGen REST API supports HTTP GET and POST requests. The MashZone NextGen API does *not* support HTTP PUT or DELETE requests.

Requests may be anonymous (a guest user with no identity) or authenticated, but are limited to those actions permitted to guests or the specific user. See "Handle Authentication" on page 1509 for details. You can "Invoke Mashables or Mashups" on page 1510 or work with snapshots (see "MashZone NextGen Snapshot API" on page 1517 for details).

Handle Authentication

Requests can be *anonymous* to have MashZone NextGen treat the user as a guest. To send anonymous requests, include the `x-p-anonymous`MashZone NextGen header/parameter in the URL or as an HTTP header.

In all other cases, the first request to MashZone NextGen must supply user identity information. This creates a standard HTTP session. What is required in the initial request depends on how authentication is configured in MashZone NextGen.

For authentication with SSO (single sign-on solution) or SSL with digital certificates, user identity information is supplied by those solutions. No additional authentication information is required in the request.

For authentication using basic credentials, you must use the REST API method to login to create a MashZone NextGen session and authenticate the user. This adds a cookie with session information which is then included in all subsequent requests. You can also use the REST API method to logout and end the session or simply let the session expire (based on MashZone NextGen timeout configuration).

- *Login*: use this URL:

  `http://`*app-server* `:`*port* `/presto/edge/api/rest/UserManagerService/login?presto_username=`*username* `&presto_password=`*pw*

  Both parameters must be in clear text.

  Successful logins set a session cookie and return an XML response in the form:

```
<?xml version="1.0"?>
<ServiceResponseObject>
  <version>2.0</version>
  <appId></appId>
  <sid>UserManagerService</sid>
  <svcVersion>1.0</svcVersion>
  <oid>login</oid>
  <response
>
    <authToken>Token is set as HTTP cookie</authToken>
  </response>
  <errorCode></errorCode>
  <invId></invId>
  <header/>
  <serviceHeaders/>
```

```
</ServiceResponseObject>
```

Unsuccessful logins return a simple text response: `Authentication failed.Userid:someName`.

- *Logout*: use this URL:

  `http://`*app-server* `:`*port* `/presto/edge/api/rest/UserManagerService/logout`

Invoke Mashables or Mashups

The form for URLs to invoke MashZone NextGen mashables or mashups using the REST API is:

`http://`*app-server* `:`*port* `/presto/edge/api/rest/`*mashable-mashup-id* `/`*operation-id* `[?`*input-parameters-if-any* `&`*presto-parameters-if-any*

- *mashable-mashup-id* = the ID of the mashable or mashup to run. See "Use Mashable/Mashup Technical Specs" on page 1489 for information on obtaining artifact IDs.
- *operation-id* = the ID of the operation in this mashable or mashup to run.

  Many mashables, such as syndicated web feeds, have a single operation. Mashups also have a single operation.
- *input-parameters-if-any* = add input parameters as query parameters in the URL when the mashable or mashup uses the HTTP GET action.

  `paramA=valueA&paramB=valueB`
- *presto-parameters-if-any* = optionally, add any "MashZone NextGen Headers/Parameters" on page 1511 to the URL to control processing, request specific information in the response or pass additional information to a mashable. Some of the most common MashZone NextGen parameters include:
  - `x-p-anonymous` to have this request treated as a guest user, with no authentication required.
  - `x-p-resultFormat` to have MashZone NextGen return the response in CSV, JSON or XML formats.
  - `x-p-paginateStart` and other pagination parameters to have MashZone NextGen paginate the result and return one 'page' of data.

See "Working with theMashZone NextGenREST API" on page 1480 for a basic example of using this API.

MashZone NextGen Headers/Parameters

MashZone NextGen provides custom headers/parameters to enable you to:

- Control certain aspects of request processing
- Provide information directly to a mashable
- Override defaults such as credentials for mashables

- Request specific information in the response

These processing flags can be added to requests as either HTTP headers or as parameters in a request URL. You can also add these processing flags in JUMP requests. See "MashZone NextGen Request Headers/Parameters" on page 1511 for a complete list.

Requests to run apps may also include some app-specific parameters. See "Parameters for App URLs" on page 1294 for a list.

MashZone NextGen may also return specific information as HTTP headers in responses. See "MashZone NextGen Response Headers" on page 1516 for a complete list.

Note: In version 2.7 and earlier, MashZone NextGen header/parameters were JUMP headers. JUMP is supported in 3.0 and above, although it has been deprecated. Some JUMP headers are no longer supported in versions 3.0 and above. See "Obsolete JUMP Headers" on page 1517 for more information.

MashZone NextGen Request Headers/Parameters

Request headers can be sent as HTTP headers or as parameters in request URLs via the REST API or using MashZone NextGen Connect for JavaScript (PC4JS). You may use `x-presto-`*headerName* or `x-p-`*headerName* synonymously.

Note: Some headers are only applicable to APIs or requests using the JUMP protocol.

presto_password	The password for basic credentials for this request. The value must be passed in clear text. Typically, this header should only be used, in combination with the `presto_username` header, to log in or authenticate a new principal when MashZone NextGen authentication is configured for basic credentials.
presto_username	The user name for basic credentials for this request. The value must be passed in clear text. Typically, this header should only be used, in combination with the `presto_pawwsord` header, to log in or authenticate a new principal when MashZone NextGen authentication is configured for basic credentials.
serviceHeader	**Note:** Only applicable to requests using the JUMP protocol. An object to forward as a header in a JUMP request to a mashable information source. A common

	example would be a SOAP header for a WSDL web service.
x-p-anonymous	Indicates that this request can be treated as a guest user, with no credentials.
x-p-dom	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-dow	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-enableSnapshot	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-httpBasicAuth	An object with the user credentials (username and password properties) to send to a mashable in an HTTP basic authentication header. If the mashable was registered in MashZone NextGen with a basic authentication security profile, these credentials override the default credentials.
x-p-hour	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-method	`delete` or `put` The MashZone NextGen Server does not support HTTP DELETE or PUT requests. This header indicates that this request is to delete or a save the subject of the request. **Note:** This header is *only* valid if the specific mashable supports HTTP DELETE or PUT or the MashZone NextGen API supports this header.
x-p-minute	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-ntlmAuth	An object with the user credentials (username, password and Windows domain properties) to send to a mashable. If the mashable was registered in MashZone NextGen with an NTLM security profile, these credentials override the default credentials.

x-p-paginateCount	The total number of items to include in one page of results for a paginated result set. See also `x-p-paginateStart`.
x-p-paginateRecord	An XPath expression identify the repeating node in results that defines one "item" to return in paginated result sets. **Note:** If there are no matches to this path, the full response from the mashup or mashable is returned.
x-p-paginateStart	An index number for the first item in a paginated result set to return in the response to this request. If this is 1, the request invokes the mashable or mashup and the MashZone NextGen Server caches the full result. The first *N* items are returned up to the value of `x-p-paginateCount`. If this value is greater than 1, the response returns the set of items requested from cache.
x-paginateToken	The pagination token returned by the MashZone NextGen Server that is used to coordinate multiple requests for paginated results from mashables or mashups. This is a response header returned when the mashable or mashup is first invoked. All subsequent requests for additional pages of results should include this header.
x-p-portType	The port type name to use for WSDL web services whose WSDL has multiple port types.
x-p-renameSOAPPrefixes	`true` | `false` A flag to have the MashZone NextGen Server change the namespace prefix used for the root element in SOAP responses to `presto`. All descendant elements with that namespace are also affected. This flag is false, by default. Use it to normalize the prefix for WSDL services that use dynamic namespace prefixes for each request.

x-p-repeatcount	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-repeatinterval	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.
x-p-resultContentType	The MIME type or Internet Media type, such as `text/plain; charset=UTF-8`, to use for the result. By default, the character encoding for text responses is UTF-8. For text media types, you can set the character encoding you want to receive as the result. You could change this to `text/plain; charset=ISO-8859-1`, for example, to get the result in a Latin-1 character encoding.
x-p-resultFeedType	Sets the syndicated web feed type and version that the response should be returned in: ■ `RSS_0.9` ■ `RSS_0.91` ■ `RSS_0.92` ■ `RSS_0.93` ■ `RSS_0.94` ■ `RSS_1.0` ■ `RSS_2.0` ■ `Atom_0.3` ■ `Atom_1.0` This header overrides the default normalization setting for the MashZone NextGen Server set by MashZone NextGen administrators. See "Syndicated Feeds (RSS/Atom)" on page 131 for the specific limitations and impacts of normalization.
x-p-resultFormat	Defines the format for the mashable or mashup response: ■ `csv` = comma-separated values. ■ `json` = JSON (JavaScript Object Notation). ■ `native` = the default format returns a response in the form native to the information source.

	■ `xml` = XML.
x-p-returnHeaderOnly	`true` | `false` A flag for *mashups only* to suppress invoking any mashables and just return HTTP headers, such as the `serviceURL` with the REST URL to invoke this mashable information source or mashup.
x-p-returnServiceHeaders	`true` | `false` A flag to control whether HTTP headers in the response from a mashable are included in the response from MashZone NextGen. This header also affects the `serviceURL` response header which contains the REST URL that corresponds to this request. Out of the box, this flag defaults to `true`, however MashZone NextGen administrators can disable this feature. This flag is only needed to override the default behavior defined in Mashup Server configuration. If the server does not forward mashable headers by default, setting this flag in a request will ensure that HTTP headers are sent and vice versa. The headers that can be forwarded are defined in a white list in MashZone NextGen Server configuration. To ensure that a specific header is forwarded, a MashZone NextGen administrator may need to update MashZone NextGen Server configuration.
x-p-returnServiceURLWithCredentials	`true` | `false` A flag. If true, this overrides the default behavior and returns actual user credentials in the `serviceURL` response header. The `serviceURL` response header contains the REST URL for this mashable or mashup. By default, the user credentials in this URL have references to session parameters rather than actual user credentials.
x-p-scheduler	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.

x-p-serviceEndpoint	For WSDL web services *only*, this is the URL that should override the existing endpoint for the web service in this request. Use this header to handle dynamic endpoints for WSDL services.
x-p-sessionbinding	An object with one or more tokens that define portions of the response to this request that should be saved in the session. Later requests can use these tokens as input parameters to be resolved from session data. Session binding tokens for this header are in the form: `@sessionbinding.attribute-name=XPath/to/nodes/to/save`
x-p-sharepoint*	MashZone NextGen headers beginning with this prefix are reserved for internal use with MashZone NextGen Add-On for SharePoint.
x-p-snapshotName	Used in the Snapshot API. See "Schedule Snapshot Job Methods" on page 1523 for information.

MashZone NextGen Response Headers

MashZone NextGen returns some information in response headers, typically when requested by headers in the request.

serviceURL	The REST URL that corresponds to the request that generated this response. This URL invokes the same mashable or mashup, with any relevant input parameters or MashZone NextGen headers. By default, this URL does not include user credentials unless specifically requested with an optional MashZone NextGen header in the request.
serviceHeader	**Note:** Only applicable to responses using JUMP. An object containing headers returned by the mashable, such as SOAP headers.
snapshotToken	The ID of the snapshot that was created from this request. See "Take or Delete Snapshot Methods" on page 1518 for more information.
x-paginateToken	The pagination token that identifies the result set for this response. The MashZone NextGen Server

	uses this token to identify which cached results to use when additional pages of results are requested. All subsequent requests should include this token as a header or parameter.
x-presto-cache	Indicates that this response has been supplied from the MashZone NextGen Response Cache.
x-totalCount	The total number of items in the cached result set for this pagination token.

Obsolete JUMP Headers

The following optional JUMP headers from versions 2.7 and earlier are obsolete and no longer supported in versions 3.0 and above:

authToken	An authentication token for requests from clients that have already logged in. Authentication has changed in version 3.0 and above to use a standard HTTP session.
password	See presto_password.
username	See presto_username.

MashZone NextGen Snapshot API

Users can take snapshots from artifact pages or schedule jobs to take snapshots automatically. These snapshots can be registered as mashable information sources and used individually to create basic apps. (See "Snapshot Mashables" on page 130 for links to topics on user functions that create or use snapshots.)

In many cases, however, snapshots are meant to be used as a set, showing trends or other statistics over a period of time or over other variables. To include this information in a mashup or custom app, you use the Snapshot API to retrieve snapshots.

This is a REST API that builds on the MashZone NextGen REST API. You can use these API methods:

- Take or Delete Snapshot Methods
- Get Snapshot Methods
- Schedule Snapshot Job Methods and Manage Scheduled Snapshot Job Methods

Take or Delete Snapshot Methods

There are two different methods that you can use to take snapshots programmatically:

- Take a Snapshot Immediately
- Take a Temporary Snapshot and Save Separately

You can also Delete Snapshots.

Take a Snapshot Immediately

You can take an immediate snapshot for any mashable or mashup using the MashZone NextGen REST URL to run the mashable or mashup plus two MashZone NextGen headers: `x-p-enableSnapshot` and `x-p-snapshotName`.

For example:

`http://`*app-server:port* `/presto/edge/api/rest/`*service* */operation* `?`*service-params-if-any* `&x-p-enableSnapshot=true&x-p-snapshotName=`*name-or-uuid*

You can find the REST API URL for any mashables or mashup in the Technical Specs tab on its artifact page. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information. Add any input parameters needed to run this mashable or mashup or any other "MashZone NextGen Headers/Parameters" on page 1511, as needed.

Parameters

Parameter	Req?	Description
`x-p-enableSnapshot`	yes	`true` = determines that the response for the mashable or mashup should create a snapshot.
`x-p-snapshotName`	yes	Choose either: ■ A specific name for this snapshot. ■ `uuid`, to have MashZone NextGen automatically assign a unique ID to this snapshot.

Take a Temporary Snapshot and Save Separately

With this pattern, you create a temporary snapshot when you run a mashable or mashup and choose to save it permanently in a separate request.

Note: Temporary snapshots must be saved within 25 minutes of their capture. After that, the MashZone NextGen Server discards the temporary snapshot.

You use the MashZone NextGen REST URL to run the mashable or mashup, but include only the `x-p-enableSnapshot`MashZone NextGen header. For example:

`http://`*app-server:port* `/presto/edge/api/rest/`*service* `/`*operation* `?`*service-params-if-any* `&x-p-enableSnapshot=true`

Add input parameters to the URL or a request body, as needed, plus any other MashZone NextGen headers you need. You can find the REST API URL for any mashables or mashup in the Technical Specs tab on its artifact page. See "Use Mashable/Mashup Technical Specs" on page 1489 for more information.

MashZone NextGen assigns a snapshot ID to the temporary snapshot and returns this ID in the `snapshotToken` HTTP header of the response, such as this example:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
...
cache-control: private, max-age=1200
serviceURL: http://localhost:8080/presto/edge/api/rest/YahooWeatherREST/getData?p%3D94572
snapshotToken: 7ac5fe05-2e67-11e0-952d-adff03bd533b
...
```

When you are sure you want to save this as a permanent snapshot, send an HTTP POST request to this URL:

`http://`*app-server:port* `/presto/edge/api/rest/snapshot`

Use the following POST body with the snapshot ID returned when the snapshot was taken and the appropriate metadata for this snapshot.

POST Body

```
<snapshot xmlns="http://www.jackbe.com/2009-08-01/snapshot"
  id="a2da57e1-2a60-11e0-9f1b-69cef3c615e5">
  <name>mySnapshot</name>
  <service>mashableA</service>
  <operation>operationName</operation>
  <createdby>userA</createdby>
  <description>Snapshot of some kind of data...</description>
  <tags>
    <tag>tag1</tag>
    <tag>tag2</tag>
    <tag>tag3</tag>
  </tags>
</snapshot>
```

Delete Snapshots

To delete a snapshot, use a URL in this form:

`http://`*app-server:port* `/snapshot/`*snapshot-id* `?x-presto-method=DELETE`

This uses the MashZone NextGen header/parameter `x-p-method`. See "Get Snapshot Methods" on page 1520 for more information on obtaining snapshot IDs.

Get Snapshot Methods

There are two query methods you can use to retrieve snapshots:

- Get Snapshot Collections By Criteria

- Get One Snapshot By ID

Typically, you must query for snapshots based on criteria to get the ID(s) for the snapshot(s) you want to retrieve. Then retrieve each snapshot based on its ID.

Get Snapshot Collections By Criteria

Base URL

To retrieve all snapshots, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/snapshots`

In most cases, you should add Parameters to filter the response to a specific set of snapshots. You can also use these parameters to sort the snapshot references in the response See Snapshot Collection Query Examples for some common combinations.

The response when you find a collection of snapshots contains a link with the URL to retrieve each matching snapshot. See Snapshot Collection Responses for an example.

Parameters

You can add any combination of the following parameters to queries for snapshot collections except that you must specify `service` if you specify `operation`.

Parameter	Req?	Description
`name`		A full or partial snapshot name. Snapshots whose name begins with this value are included.
`tag`		A tag to match.
`service`		The name of the mashable or mashup that this snapshot was generated from. You can use this parameter by itself or in conjunction with the `operation` parameter. The name is case sensitive. You can also use partial artifact names. This matches artifacts whose name begins with the value you specify.
`operation`		The operation for the mashable or mashup that this snapshot was generated from. You must use this parameter in conjunction with the `service` parameter.
`createdby`		The name of the user who generated this snapshot.

Parameter	Req?	Description
`fromdate`		The first scheduled execution date to include in the form: `YYYY-MM-DD` See Snapshot Collection Query Examples for an example.
`todate`		The first scheduled dateto exclude in the form: `YYYY-MM-DD` See Snapshot Collection Query Examples for an example.
`sortby`		The field to use to sort snapshot references by in the response. You can sort by: ■ `name` ■ `tags` ■ `service` ■ `operation` ■ `createdby` (default)
`sortdir`		The sort order to use, if you use the `sortby` parameters. This can be `ascending` (default) or `descending`.

Snapshot Collection Query Examples

Get snapshots by name or partial name:

```
http://localhost:8080/presto/edge/api/rest/snapshots?name=mySnap
http://localhost:8080/presto/edge/api/rest/snapshots?name=my
```

Get all snapshots for all operations of one service:

```
http://localhost:8080/presto/edge/api/rest/snapshots?service=YahooWeatherREST
```

Get all snapshots for one operation of a service:

```
http://localhost:8080/presto/edge/api/rest/snapshots?service=XigniteHistorical&operation=GetTopMo
```

Get all snapshots created by one user:

```
http://localhost:8080/presto/edge/api/rest/snapshots?createdby=JAgusi
```

Get all snapshots for a tag and sort by artifact name:

```
http://localhost:8080/presto/edge/api/rest/snapshots?tag=news&sortby=service
```

Get all snapshots created since a specific start date:

```
http://localhost:8080/presto/edge/api/rest/snapshots?fromdate=2011-01-31
```

Get all snapshots created on one specific date:

```
http://localhost:8080/presto/edge/api/rest/snapshots?fromdate=2011-01-31&todate=2011-02-01
```

Get all snapshots created by one user using a specific tag

```
http://localhost:8080/presto/edge/api/rest/snapshots?createdby=ladams&tag=projectA
```

Snapshot Collection Responses

Snapshot collections contain metadata for matching snapshots but no snapshot data. This metadata includes the snapshot ID as well as the full URL (in `content/@href`) that you can use to retrieve individual snapshots.

```
<snapshots xmlns="http://www.jacbe.com/2009-08-01/snapshot">
  <snapshot id="9445ae9b-28d3-11e0-9c7b-f391b2692a54">
    <name>Snapshot - Tue Jan 25 2011</name>
    <service>XigniteHistorical</service>
    <operation>GetTopMovers</operation>
    <createdby>Jane</createdby>
    <description>10 top movers 12/31/2010</description>
    <tags><tag>movers</tag></tags>
    <view>Grid</view>
    <createdtime>2011-01-25T14:37:41.935-08:00</createdtime>
    <createdtimemillis>1295995061960</createdtimemillis>
    <annotations count="0"/>
    <content type="uri"
href="/presto/edge/api/rest/snapshot?id=9445ae9b-28d3-11e0-9c7b-f391b2692a54"
/>
  </snapshot>
  <snapshot id="7f6cb381-298e-11e0-afe3-8dcf3e37fbe1">
    ...
  </snapshot>
  ...
</snapshots>
```

Get One Snapshot By ID

Base URL

To retrieve a specific, complete snapshot, you must have the snapshot ID. Typically, you find this ID in the response to a snapshot collection query. See "Get Snapshot Collections By Criteria" on page 1520 for an example of this response.

You use the URL in `content/@href` from the snapshot collection query response to retrieve the snapshot you want. This URL is in the form:

`http://`*app-server:port* `/presto/edge/api/rest/snapshot/`*snapshot-id*

A Single Snapshot Response

This example shows a snapshot in XML format. The actual results in the snapshot are contained in the `content` element:

```
<snapshot xmlns="http://www.jackbe.com/2009-08-01/snapshot"
    id="747523af-2e50-11e0-bde1-61d07a0308ed">
  <name>Top Losers Dec 31, 2010</name>
```

```
  <service>XigniteHistorical</service>
  <operation>GetTopLosers</operation>
  <createdby>admin</createdby>
  <description>Top stock lowers for 12/31/2010</description>
  <tags>
    <tag>losers</tag>
  </tags>
  <view>Grid</view>
  <createdtime>2011-02-01T14:16:07.928-08:00</createdtime>
  <createdtimemillis>1296598567932</createdtimemillis>
  <content xmlns="">
     <!-- actual snapshot data -->
    <GetTopLosersResponse xmlns="http://www.xignite.com/services/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
     <GetTopLosersResult>
     ...
    </GetTopLosersResponse>
  </content>
  <annotations/>
  <filter/>
</snapshot>
```

Schedule Snapshot Job Methods

Base URL

You can schedule a job to take snapshots at regular intervals for any mashable or mashup. Use the MashZone NextGen REST URL to run the specific mashable or mashup as the base URL to schedule a job. You can find this URL in the Technical Specs tab on the artifact page for any mashables or mashup.

Include input parameters, if any, for the mashable or mashup in the URL and any other MashZone NextGen headers, as needed.

Then add the `x-p-enableSnapshot` and `x-p-snapshotName` parameters. For example:

`http://`*app-server:port* `/presto/edge/api/rest/`*service* */operation* `?`*service-params-if-any* `&x-p-enableSnapshot=true&x-p-snapshotName=`*name-or-uuid*

To this base URL you add appropriate Parameters that define the schedule for the jobs.

Important: It is a best practice to limit snapshot schedules to a specific number of repetitions, although this is not required. Open-ended schedules can have a negative impact on performance.

See Schedule Examples for some combinations of these parameters.

Parameters

You must specify the schedule for the job and any additional parameters required for that type of schedule.

Parameter	Req?	Description
`x-p-enableSnapshot`	yes	Determines that this invocation is to create a scheduled job.
`x-p-snapshotName`	yes	Choose either: ■ A single name to assign to all snapshots taken by this job. ■ `uuid`, to have MashZone NextGen automatically assign a unique ID to each snapshot for this job.
`x-p-scheduler`	yes	The type of schedule for this job. Valid schedule types include: ■ `secondly` = schedule every second. Or schedule every number of seconds defined in `x-p-repeatinterval` with a schedule termination in `x-p-repeatcount`. ■ `minutely` = schedule every minute. Or schedule every number of minutes defined in `x-p-repeatinterval` with a schedule termination in `x-p-repeatcount`. ■ `hourly` = schedule every hour. Or schedule every number of hours defined in `x-p-repeatinterval` with a schedule termination in `x-p-repeatcount`. ■ `daily` = schedule every day at a specific time. You must include the `x-p-hour` and `x-p-minute` parameters with a schedule of this type. ■ `weekly` = once a week, on a specific day of the week at a specific time. You must include the `x-p-dow`, `x-p-hour` and `x-p-minute` parameters with a schedule of this type. ■ `monthly` = once a month, on a specific day at a specific time. You must include the `x-p-dom`, `x-p-hour` and `x-p-minute` parameters with a schedule of this type.
`x-p-hour`		The hour, from 0 to 23, to run a daily, weekly or monthly scheduled job. You must also add the `x-p-minute` parameter.

Parameter	Req?	Description
`x-p-minute`		The minute to run a daily, weekly or monthly scheduled job. You must also add the `x-p-hour` parameter.
`x-p-dow`		The day of the week, from 1-7 beginning on Sunday, to run a weekly scheduled job. You must also add the `x-p-hour` and `x-p-minute` parameters.
`x-p-dom`		The day of the month (1-31) to run a monthly scheduled job. You must also add the `x-p-hour` and `x-p-minute` parameters.
`x-p-repeatcount`		The maximum number of scheduled snapshots to take for this scheduled job. You must also use the `x-p-repeatinterval` parameter. **Tip:** Is it highly recommended to include this optional parameter in job schedules that use second, minute or hourly intervals to define a termination point for the job.
`x-p-repeatinterval`		The number of time measurements between each scheduled snapshot for a scheduled job. The actual time measurement is defined by the value of the `x-p-scheduler` parameter, such as `hourly`. You must also specify the `x-p-repeatcount` parameter to define a limit for the schedule.

Schedule Examples

These examples all use the sample mashable Yahoo Weather Given Zipcode which is a REST mashable that takes a single input parameter, `p`, with a zip code:

- Schedule jobs every 3 hours for a 24 hour period:

  `http://localhost:8080/presto/edge/api/rest/YahooWeatherREST/getData?p=98106&x-p-enableSnapshot`

- Schedule a job on Wednesday each week at 6:00p.m., with no termination:

  `http://localhost:8080/presto/edge/api/rest/YahooWeatherREST/getData?p=98106&x-p-enableSnapshot`

Manage Scheduled Snapshot Job Methods

You can retrieve information about scheduled snapshot jobs (see "Find Jobs" on page 1526) or manage jobs (see "Suspend Jobs" on page 1526, "Resume Jobs" on page 1527 and "Delete Jobs" on page 1527).

Find Jobs

To retrieve information about scheduled snapshot jobs, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/jobs?`

Add the following parameters as needed to define the scope for this query.

Parameter	Req?	Description
`user`	yes	The name of the user whose job schedules should be returned.
`service`	yes	The name of the mashable or mashup to retrieve job information for.
`operation`		An optional name of the specific operation for this mashable or mashup to retrieve job information for.

This example retrieves jobs for a specific mashable operation and user:

```
http://localhost:8080/presto/edge/api/rest/meta/jobs?user=sdiggle
&service=employees-table&operation=findByID
```

This example retrieves jobs for all operations of a specific mashable and user:

```
http://localhost:8080/presto/edge/api/rest/meta/jobs?user=admin
&service=XigniteHistorical
```

Suspend Jobs

To suspend the scheduled execution of all snapshot jobs, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/suspendall`

To suspend the schedule execution of a specific snapshot job, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/suspendjob?user=`*username* `&job=`*jobname*

Both the `user` and `job` parameters are required.

Resume Jobs

To resume the scheduled execution of all snapshot jobs, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/resumeall`

To resume the schedule execution of a specific snapshot job, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/scheduler/resumejob?user=`*username* `&job=`*jobname*

Both the `user` and `job` parameters are required.

Delete Jobs

To delete scheduled executions of a snapshot job, use a URL in the form:

`http://`*app-server:port* `/presto/edge/api/rest/meta/job/`*username* `/`*jobname* `?x-presto-method=DELETE`

This uses the MashZone NextGen header/parameter `x-p-method`.

About DataTable Events

Mashable and mashup results used with views in basic apps are held in a *DataTable*, a client-side data model. See the MashZone NextGen DataTable API for details.

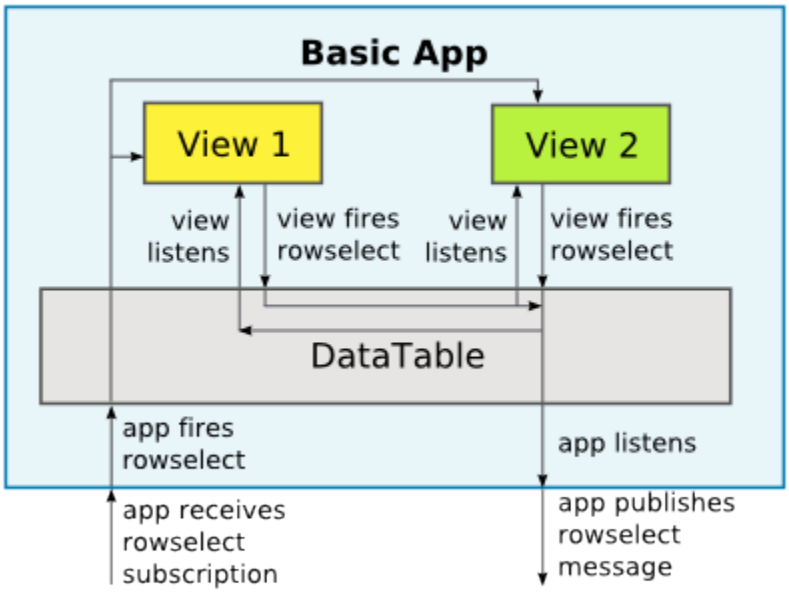
Each basic app has a DataTable with results for their mashable or mashup. This includes basic apps created in App Maker and views that have been directly added to workspace apps if they are configured to share data (see "Automatic Interactions and Shared Properties for Multiple Views Directly Added to a Workspace" on page 1125).

Built-in and pluggable views included in a basic app interact with result data through the app's DataTable. Custom apps can also use DataTables, if desired.

In addition to basic data access, DataTables support result pagination, track selection state in result rows and trigger events for row selection. DataTables use a single-row selection model.

DataTable Events for Views and Apps

DataTables have a single event, `rowselect`, of interest to views and basic apps. This event indicates that one row of data in results has been selected by the user, typically through the user interface of a view within the app. If the app is part of a workspace, however, this change in row selection can also come from messages published by another app that are wired to the `rowselect` subscription topic.



Views can both trigger and listen for DataTable `rowselect` events, allowing all interested views that are using a DataTable to synchronize their selection state across the app. Basic apps also listen for `rowselect` events in the DataTable and then publish messages to the `rowselect` topic or trigger `rowselect` events in the DataTable as the result of messages received from other apps in the `rowselect` subscription topic. Subscriptions to `rowselect` topics can also be used to synchronize app state when the workspace initially loads (see "Synchronizing Wired Apps When a Workspace Loads" on page 1181 for information).

Note: Wiring in a workspace app that ties `rowselect` events in the DataTable for one app to row selection for DataTables in other apps is possible, but may not always be consistent. Pagination in a DataTable can cause matches to be missed. Multiple matches will also not be found as the selection model for the DataTable is a single selection.

MashZone NextGen built-in views that support some type of 'click' event both trigger and listen for DataTable `rowselect` events. Pluggable views can also choose to trigger or listen for this event. See "Pluggable View Classes: Triggering and Handling Events" on page 1059 for more information.

A `rowselect` event includes these properties:

- `rowIndex` = the index number of the row for this event
- `selected` = a boolean indicating whether this row should be selected (true) or not (false)
- `rowData` = a row object with the data for this row

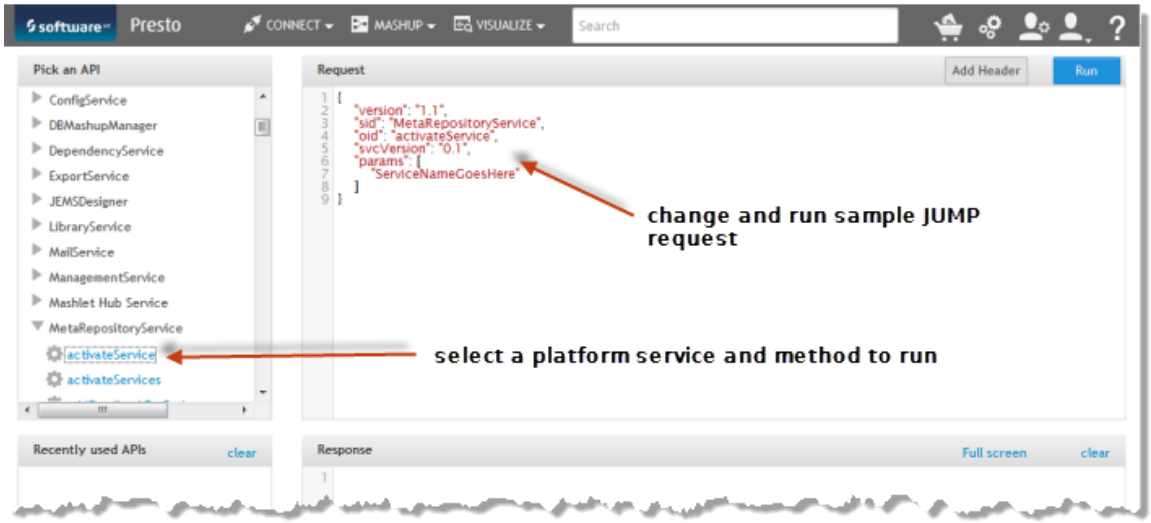
When a view indicates a row has been clicked, the DataTable fires one or two `rowselect` events:

- For the existing selected row, if any, an event with `selected = false` to clear selection for this row.
- If the clicked row is not the existing selected row, an event with `selected = true` to set the selected state for this new row.

MashZone NextGen Platform API Console

This console allows MashZone NextGen administrators and developers to discover and run platform APIs for MashZone NextGen. Platform APIs can be useful to investigate issues that you may report to Technical Support. They may also be used in some cases during upgrades to a new version of MashZone NextGen. Platform APIs can also be used to provide specialized features or integration.

Click in the main menu to open the API Console:



Platform APIs are organized in services that focus on specific aspects of MashZone NextGen. Find and select the MashZone NextGen Platform service and method you want to run. In most cases, the service names are self-explanatory. See "MashZone NextGen Platform Services" on page 1530 for more information if needed.

Once you select a platform API method, a sample request to invoke this method fills the Request pane.

Note: Most platform APIs use the JUMP (JackBe Universal Message Protocol) format. See "The JackBe Universal Message Protocol (JUMP)" on page 1531 for more information on this format.

Click **Run** to send the request. The JUMP response from the MashZone NextGen Server displays in the Response pane. See "JUMP Responses" on page 1532 for more information on responses.

MashZone NextGen Platform Services

Service	Description
AppSearch	Contains methods to find apps or workspace apps based on a variety of characteristics.
AppService	Contains methods to manage apps and workspace apps including add, remove, activate, publish and so on. This service also provides methods to get app specifications or specific portions/properties of app specifications.
AppStoreService	Contains methods to manage apps in the AppDepot including submit, approve, reject, add to favorites and so on. Also contains methods to find apps published to AppDepot based on various characteristics.
CollabService	Contains methods around collaboration in MashZone NextGen Hub including activity, commenting, broadcast messages, rating, usage and so on.
ConfigService	Contains methods to get and set configuration properties for the MashZone NextGen Server. Includes both methods for specific properties and generic methods to get and set properties by name.
DBMashupManager	Contains methods to manage drivers, datasources and the database services based on registered datasources.
DependencyService	Contains methods to track and manage dependencies between MashZone NextGen artifacts.
ExportService	
JEMSDesigner	Contains methods to manage macros and macro domains.
LibraryService	Contains methods to find and manage libraries and other resources used in pluggable views.
MailService	Contains methods used to manage and send email notifications for sharing.

Service	Description
ManagementService	Contains methods to find license and statistics information for the Admin Console.
MetaRepositoryService	Contains methods to manage mashables, mashups, categories, providers and MashZone NextGen global attributes including add, register, activate, find, validate and so on.
PolicyService	Contains methods to find and manage permission sets for run permissions, other artifact actions and access to specific features (entitlements) in MashZone NextGen Hub and the AppDepot. Also contains methods to create and manage dynamic groups for custom policies and manage policy configuration.
ProfileService	Contains methods to find and manage user profiles for Mashboard. Mashboard profiles save the state of each user's currently open workspaces.
ResourceService	Contains methods to find and manage file resources used in apps, pluggable libraries, custom Wires blocks or other extensions to MashZone NextGen.
Sample Requests with Test Data	A set of sample requests to invoke various mashables.
ShareManager	No longer used.
SharePointManager	Contains methods to get connections, sites, lists and other configuration used to register Lists or use SharePoint as a source of information in mashables or mashups.
UserManagerService	Contains methods to find and manage users, groups and MashZone NextGen users attributes.

14 MashZone NextGen Administration

- Getting Started with the MashZone NextGen Server 1563
- MashZone NextGen Security 1602
- MashZone NextGen Server Configuration 1626
- Integrated MashZone Server Configuration and Administration 1696
- Event Service Configuration and Administration 1704
- Process Performance Manager (PPM) Integration 1732
- MashZone NextGen and MashZone Repositories 1737
- MashZone NextGen Server Administration 1740

MashZone NextGen may be installed by itself or as part of Intelligent Business Operations (IBO).

MashZone NextGen administrators install, configure and manage the:

- MashZone NextGen Server
- Integrated MashZone Server
- Event Service
- MashZone NextGen Repository and MashZone Repository

See the installation and administration documentation for more information.

The following table lists starting points to help you find the tasks and configuration options for administrators:

Post-installation tasks	See "Getting Started with the MashZone NextGen Server" on page 1563 for required tasks. See "What's Next" on page 1601 for links to optional configuration tasks for the MashZone NextGen Server and IBO.
Overview	See "What is Installed with MashZone NextGen" on page 1569.
Authentication, authorization and other security configuration	See "MashZone NextGen Security" on page 1602 for concepts and configuration tasks.
Other MashZone NextGen Server configuration tasks	See "MashZone NextGen Server Configuration" on page 1626 and "Process Performance Manager (PPM) Integration" on page 1732 for links to a variety of configuration tasks.
Integrated MashZone Server and Event Service configuration	See "Integrated MashZone Server Configuration and Administration" on page 1696 and "Event Service Configuration and Administration" on page 1704 for more information.
MashZone NextGen Repository configuration	"MashZone NextGen and MashZone Repositories" on page 1737

Administration, upgrade and deployment tasks	"MashZone NextGen Server Administration" on page 1740

Getting Started with the MashZone NextGen Server

You install MashZone NextGen using the Software AG Installer. See the *Installing Software AG Products* guide for instructions.

The post-installation tasks you must complete to allow users to start working with MashZone NextGen include:

1. Start the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Login to MashZone NextGen Hub:
   a. Open MashZone NextGen Hub in a browser at `http://localhost:8080/presto`.

      If you used a different port number when you installed MashZone NextGen or the MashZone NextGen Server is running on a different host, change the domain and port number appropriately.

   b. Use the credentials for the default administrator account:
      - User name = `Administrator`
      - Password = `manage`
3. If you are using the default MashZone NextGen User Repository rather than an LDAP Directory to manage users and groups for MashZone NextGen, it is a good practice to change the password for this default administrator account.
   a. Open your profile from the MashZone NextGen Hub menu bar and click **My Password**.
   b. Enter your new password and confirm this.
   c. Then click **Change Password**.

   If you will use your LDAP Directory as the MashZone NextGen User Repository, this default account is disabled once LDAP configuration is complete.
4. Set up a robust database to use as the MashZone NextGen Repository.

   MashZone NextGen is installed with Derby as an embedded database hosting the MashZone NextGen Repository for trial purposes only. The default Derby database should *not* be used for serious development environments or for staging or production.

   See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for instructions.

For MashZone NextGen database compatibility information, see "MashZone NextGen and MashZone Database Compatibility" on page 1567.

5. If you want MashZone NextGen to use your LDAP Directory as the repository for user and group information, you must update configuration. See "Integrate Your LDAP Directory with MashZone NextGen" on page 1590 for instructions.
6. Configure the Integrated MashZone Server and the Event Service. See "Integrated MashZone Server Configuration and Administration" on page 1696 and "Event Service Configuration and Administration" on page 1704 for instructions.
7. If you have also installed Terracotta BigMemory and received your BigMemory license, add this license to MashZone NextGen and configure MashZone NextGen to work with BigMemory. See "Manage Licenses for MashZone NextGen, Universal Messaging and BigMemory" on page 1572 and "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for instructions.

In addition to these required steps, there are many other optional configuration tasks you may wish to complete. See "What's Next" on page 1601 for links to optional tasks.

Additional MashZone NextGen System and Software Requirements

- For basic requirements to install MashZone NextGen, see the *System Requirements for Software AG Products* guide. For additional requirements or requirements for MashZone NextGen Add-Ons, see:
  - "Additional Recommendations for MashZone NextGen" on page 1564
  - "Recommendations for MashZone NextGen Add-On for SharePoint (P4S)" on page 1565
  - "Recommendations for the MashZone NextGen Add-On for Excel" on page 1566
  - "Recommendations for the MashZone NextGen Add-On for Portals" on page 1566
  - "Requirements for Optional Views" on page 1566
- For information on other compatibility questions, see:
  - "MashZone NextGen Browser and Mobile Device Compatibility" on page 1567
  - "MashZone NextGen and MashZone Database Compatibility" on page 1567

Additional Recommendations for MashZone NextGen

In addition to the basic recommendations in the *System Requirements for Software AG Products* guide, you should also consider the following recommendataions for MashZone NextGen:

- A robust, compatible database to host the MashZone NextGen Repository is required.

Important: TheMashZone NextGenRepository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution. See "MashZone NextGen and MashZone Database Compatibility" on page 1567 for more information.

- Architecture and memory requirements or recommendations include:
  - 64-bit architecture
  - 1G minimum of memory if only small to medium datasets are involved
  - 4G minimum of memory if large datasets are involved

Important: Actual memory and disk space requirements vary widely based on load, throughput, performance and other requirements unique to your environment. Please contact your Software AG representative for more information.

See also "MashZone NextGen Browser and Mobile Device Compatibility" on page 1567.

Recommendations for MashZone NextGen Add-On for SharePoint (P4S)

- A compatible version of SharePoint:

MashZone NextGen Add-On for SharePoint / MashZone NextGen	Microsoft Office SharePoint 2007	Microsoft SharePoint Server 2010
P4S 3.0 + MashZone NextGen 3.0	yes	
P4S 3.1 + MashZone NextGen 3.1		yes
P4S 3.1.1 + MashZone NextGen 3.1.1 or greater	yes	yes

The Single Sign-on Service (2007) or Secure Store Service (2010) in SharePoint are required to use the MashZone NextGen Add-On for SharePoint Single Sign-On feature. This significantly improves the user experience by removing or reducing login challenges. See Microsoft documentation for more details on single sign-on capabilities in SharePoint.

- Microsoft .NET Framework 3.5 (available at http://www.microsoft.com/downloads/details.aspx?FamilyId=333325fd-ae52-4e35-b531-508d977d32a6).
- An appropriate version of the Windows operating system and hardware. See Microsoft documentation for details.

  Note: Per Microsoft, SharePoint Server 2010 requires a 64-bit architecture.

- MashZone NextGen Add-On for SharePoint supports SharePoint 2010 with Firefox, Chrome, and MS Internet Explorer 10 and higher. If you want to use the SharePoint add on with SharePoint 2010 and Internet Explorer Versions 7, 8 and 9 you have to force SharePoint 2010 to use the Internet Explorer 10 mode. For this you need to update your master page to set the version of Internet Explorer to 10 or above using an appropriate meta tag, e.g. <meta http-equiv="X-UA-Compatible" content="IE=10" />.

In most installations, authentication between SharePoint and MashZone NextGen uses the NTLM version 2 protocol.

Recommendations for the MashZone NextGen Add-On for Excel

- 1.7 megabytes (MB) of available hard disk space.
- Recommended hardware requirements for Windows Office 2007 or 2010 and the .NET Framework 2.0 or above.
- Microsoft Office 2007 or Office 2010 (for a 32 bit architecture) and any compatible Windows operating system.

  Note: Because of a known issue, the MashZone NextGen Add-On for Excel is not currently compatible with Microsoft Office 2010 for 64 bit systems.

- Windows Office 2007 Primary Interop Assemblies.
- Microsoft .NET Framework 2.0 or above.

Recommendations for the MashZone NextGen Add-On for Portals

The MashZone NextGen Add-On for Portals is a portlet producer that is compatible with any portal that supports the JSR-168 Portlet Specification or JSR-286 Portlet 2.0 Specification. The MashZone NextGen Add-On for Portals has no additional system requirements beyond those for your portal.

Requirements for Optional Views

Google Map views do not require an API key from Google, unless your usage exceeds Google limitations.

MashZone NextGen Browser and Mobile Device Compatibility

This release of MashZone NextGen has been tested against the following browsers and devices. MashZone NextGen tools and apps are compatible with the known exceptions noted here. See also "Requirements for Optional Views" on page 1566 for additional requirements.

Desktop	Mobile Device and OS
Internet Explorer 9.0 and 10. There is one known exception. MashZone NextGenMashboard, the visual composer for workspace apps, automatically runs in IE8 forced compatibility mode when running in IE 9.0.	**Note:** Currently MashZone NextGen Mobile apps, the 'mobile editions' of the AppDepot, are not available for Android devices. However, mobile-compatible MashZone NextGen apps can be viewed through the browser on Andriod devices.
Chrome	iPhone or iPad devices with iOS 4.3.3 and above
Firefox 24.0	
Safari 7	

MashZone NextGen and MashZone Database Compatibility

MashZone NextGen has an internal database, the MashZone NextGen Repository. The Integrated MashZone Server for MashZone NextGen also has an internal database. Both of these repositories are installed in separate Derby databases that are configured as *embedded* databases running in the same JVM as MashZone NextGen and the Integrated MashZone Server.

Important: TheMashZone NextGenRepository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

These repositories are compatible with the following databases and drivers:

Database / Driver	MashZone NextGen	MashZone
Apache Derby 10.5.3 (embedded)	✓	✓

Database / Driver	MashZone NextGen	MashZone
derby-embedded-10.5.3.jar		
Microsoft SQL Server 2008 (10.00.2531) jtds-1.3.1.jar	✓	✓
Microsoft SQL Server 2012 (11.0.2012) jtds-1.3.1.jar		✓
MySQL 5 mysql-connector-java-5.1.6-bin.jar	✓	
MySQL 5.6.1 mysql-connector-java-5.1.6-bin.jar	✓	✓
Oracle 11g (11.1.0.6) ojdbc16.jar **Note:** Scripts are compatible with the Oracle SQL Plus client.	✓	✓
Oracle 12c (12.1.0.6) ojdbc16.jar **Note:** Scripts are compatible with the Oracle SQL Plus client.	✓	✓
PostgreSQL 9.0 postgresql-9.0-801.jdbc3.jar	✓	
PostgreSQL 9.3.1 postgresql-9.2-1002.jdbc4.jar		✓

Both MashZone NextGen and MashZone also work with databases making database assets visible in MashZone NextGen as mashable information sources or MashZone feeds. Database mashables have been tested with:

- DB2 9.7.400.501 (db2jcc.jar)
- Microsoft SQL Server 2008 (jtds-1.2.5.jar)
- MySQL 5.x (mysql-connector-java-5.1.6-bin.jar)
- Oracle 11g (ojdbc14.jar)
- PostgreSQL 9 (postgresql-9.0-801.jdbc3.jar)

What is Installed with MashZone NextGen

MashZone NextGen initially installs these WAR files:

WAR	Server and/or Application
presto.war	MashZone NextGen Server, MashZone NextGen Hub and AppDepot
mashzone.war	Integrated MashZone Server
rtbs.war	Event Service
ibo.war and ibo-config.war	IBO user interface

Note: You must download some MashZone NextGen Add-Ons separately. See Add-ons, Updates, Forums and Technical Support for more information.

MashZone NextGen also installs the following additional software:

- Apache's TomEE Plus Servlet Container, version 7.0.47
- Derby Database, version 10.5.3.0.

Important: TheMashZone NextGenRepository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution. See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for details.

Finally, MashZone NextGen includes installation packages for built-in apps and dashboards that you can choose to import to make them available to users. This can include:

- The PPM Chart custom app to allow users to easily pull charts from PPM into workspace apps (dashboards).

- Workspace apps (dashboards) for business process monitoring from Optimize.

MashZone NextGen Installation Folders

```
SoftwareAG
- MashZoneNG
  - apache-tomee-jaxrs (the container and web apps)
  - ibo-dashboards (for IBO + built-in dashboards)
  - mashzone (external resources for the Integrated Servers)
    - clientapps (Zip for PPM Chart app)
    - data
      - importexport
      - jdbcdrivers
      - resources
    - tools
  - prestocli (MashZone NextGen command line utilities and samples)
    - bin
    - lib
    - raql-samples
    - sample (emml)
    - schemas (emml)
  - prestorepository (scripts for moving the repository)
    - derby
    - mssql
    - mysql
    - oracle
    - postgres
  - raql-udfs (for RAQL built-in and user-defined functions)
    - analytics
    - SampleRaqlLib
```

Start and Stop the MashZone NextGen Server

The MashZone NextGen Hub, AppDepot, MashZone NextGen Mobile apps and the MashZone NextGen Add-Ons for Excel, SharePoint and Portals all depend on the MashZone NextGen Server. If MashZone NextGen is installed as part of Intelligent Business Operations, IBO dashboards also depend on the MashZone NextGen Server.

The MashZone NextGen Server depends on the MashZone NextGen Repository. Start-up and shutdown for the MashZone NextGen Server also automatically starts and stops the Integrated MashZone Server and the Event Service which depend on the MashZone Repository.

See "Start the MashZone NextGen Server" on page 1570 and "Stop the MashZone NextGen Server" on page 1572 for instructions.

Start the MashZone NextGen Server

1. If you have installed the MashZone NextGen Add-On for SharePoint Add-On and the MashZone NextGen Server uses the same LDAP Directory as SharePoint, log in to the MashZone NextGen Server host using the correct startup user account. See "Startup Account for the MashZone NextGen Server With MashZone NextGen Add-On for SharePoint" on page 1571 for more information.

2. If the MashZone NextGen Repository and MashZone Repository have been moved from the default Derby database and they are *not* already running, manually start these databases following the instructions for their host database.
3. Do one of the following to start the MashZone NextGen Server:
   - For Windows systems, either:
     - From the Start menu, select **Software AG > Start Servers > Start MashZone NextGen *version***.
     - Enter this command in a command window:

       ```
       c:>MashZoneNG-install/apache-tomee-jaxrs/bin/catalina.bat start
       ```

     Note: On Windows Server operating systems MashZone NextGen Server needs to be started as Administrator. In order to run MashZone NextGen Server as Administrator right click on the **Start MashZone NextGen *version*** shortcut and select **Run as administrator**.
   - For Linux, Mac OS X or UNIX systems, open a new terminal window and move to this folder:

     ```
     % cd MashZoneNG-install/apache-tomee-jaxrs/bin
     ```

     Then enter this command:

     ```
     % catalina.sh start
     ```

   This also automatically starts the Integrated MashZone Server and the Event Service.

Open the MashZone NextGen Hub at `http://app-server:port/presto` and log in. You can now access the AppDepot and all the MashZone NextGen tools: Mashboard, Wires, the MashZone Feed Editor, the Mashup Editor, the App Editor and the Admin Console.

Startup Account for the MashZone NextGen Server With MashZone NextGen Add-On for SharePoint

There is a known issue with authentication for SharePoint users working with MashZone NextGen Add-On for SharePoint when both the MashZone NextGen Server and SharePoint use the same LDAP Directory for authentication. When users work in MashZone NextGen Hub to register SharePoint lists, use SharePoint lists or searches in mashups or publish apps to SharePoint, the credentials for the user account that started MashZone NextGen Server is used initially for authentication and authorization rather than the credentials of the current user or the configured SharePoint connection.

To work around this issue, you must use a user account that has been defined specifically to start up the MashZone NextGen Server in this context. You have two options to create this special startup account:

- Create a user account on the host for the MashZone NextGen Server that does not exist in your LDAP Directory.

- Add a user to your LDAP Directory but do *not* grant this user permissions to work in SharePoint.

Stop the MashZone NextGen Server

1. Do one of the following:
   - For Windows systems, either:
     - From the Start menu, select **Software AG > Stop Servers > Stop MashZone NextGen**.
     - Enter this command in a command window:

       ```
       c:>MashZoneNG-install/apache-tomee-jaxrs/bin/catalina.bat stop
       ```
   - For Linux, Mac OS X or UNIX systems, open a new terminal window and move to this folder:

     ```
     % cd MashZoneNG-install/apache-tomee-jaxrs/bin
     ```

     Then enter this command:

     ```
     % cataline.sh stop
     ```

   This also automatically stops the Integrated MashZone and Integrated Event Service servers.
2. If the MashZone NextGen Repository and MashZone Repository have moved from the default Derby database, you can also choose to stop these two databases. See documentation for their host database for instructions.

Startup Considerations

When the MashZone NextGen Repository is hosted in a robust database solution, it must be started before the MashZone NextGen Server for a successful startup. With the default Derby database, the MashZone NextGen Repository runs as an *embedded* database that is automatically started with the MashZone NextGen Server.

In environments where you application server is started automatically with the host, this can create timing errors. You may need to stop and restart the MashZone NextGen Server after the MashZone NextGen Repository has been restarted.

If you host the MashZone NextGen Repository in a MySQL or Oracle database, you may also be able to have the database start automatically with the host.

Manage Licenses for MashZone NextGen, Universal Messaging and BigMemory

MashZone NextGen requires two licenses: one for MashZone NextGen and one for the Integrated Universal Messaging that is deployed with MashZone NextGen.

If you are using BigMemory features that require this, you also need to make your BigMemory license available to the MashZone NextGen Server and/or the Integrated MashZone Server. See "BigMemory for Caching, Connections and MashZone NextGen Analytics" on page 1661 for features that require this additional license.

You can apply licenses when you install MashZone NextGen, or you can install and use MashZone NextGen without a license for a trial period of 30 days. If you purchase MashZone NextGen after installation, you must manually apply the MashZone NextGen and Universal Messaging licenses. If needed, you can also manually apply a BigMemory license.

Note: When MashZone NextGen runs with a READ ONLY license, all tools to create mashables, mashups and apps (App Maker, Mashboard, App Editor, Wires and Mashup Editor) are unavailable.

To manually apply any of these licenses

1. Save the attached license file(s) from the email(s).
2. For MashZone NextGen licenses, copy the MashZoneNextGenLicense.xml file into the *MashZoneNG-install* /apache-tomee-jaxrs/conf folder.

   Note: If MashZone NextGen is deployed in a cluster, copy the license file to this folder in every cluster member.

3. For the Integrated Universal Messaging license, copy the license.xml file to the *MashZoneNG-install* /um folder.

   Note: If MashZone NextGen is deployed in a cluster, copy the license file to this folder in every cluster member.

4. If a BigMemory license is required:
   a. Copy the license file terracotta.key to the *MashZoneNG-install* /apache-tomee-jaxrs/conf folder.

      Note: If MashZone NextGen is deployed in a cluster, you must copy this file to every cluster member.

   b. Add this Java system property to the MashZone NextGen startup script :

      ```
      -Dcom.tc.productkey.path=MashZoneNG-install/apache-tomee-jaxrs/
      conf/terracotta.key
      ```

      Add this property to either:

      - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat file for Windows systems, or
      - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX systems.

Note: If MashZone NextGen is deployed in a cluster, you must update the startup scripts for every cluster member.

5. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution

TheMashZone NextGenRepository and MashZone Repository are initially installed in a Derby database suitable *only for trial*purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

You can host the MashZone NextGen and MashZone Repositories in any database that MashZone NextGen or MashZone supports. See "MashZone NextGen and MashZone Database Compatibility" on page 1567 in System Requirements for details.

You can move both of these repositories to one of the following databases:

- *Microsoft SQL Server*: see "Move MashZone NextGen and MashZone Repositories to Microsoft SQL Server" on page 1575 for instructions.
- *MySQL*: see "Move the MashZone NextGen and MashZone Repositories to MySQL" on page 1579 for instructions.
- *Oracle*: see "Move the MashZone NextGen and MashZone Repositories to Oracle" on page 1582 for instructions.
- *PostGres*: see "Move the MashZone NextGen and MashZone Repositories to PostGres" on page 1586 for instructions.

Troubleshooting Connections to the MashZone NextGen Repository

The most common problem when the MashZone NextGen Server server does not restart successfully after you move the MashZone NextGen Repository to a new database is that is cannot connect to the MashZone NextGen Repository. To verify that this is the problem:

1. Open the MashZone NextGen Server log file prestoserver.log in your web application server's log directory. For TomEE, this is:

   web-apps-home /logs/prestoserver.log

2. Check for a log entry for `Cannot create PoolableConnectionFactory` near the end of the file. This error indicates the MashZone NextGen Server could not successfully connect to the new database.

Common causes for this error include:

- The database hosting the MashZone NextGen Repository is not running.

If this is true, start the MashZone NextGen Repository and verify that it is up. Then restart the MashZone NextGen Server and confirm that this starts successfully.

- There are one or more firewalls between the MashZone NextGen Repository and the MashZone NextGen Server that are not configured to allow this connection.

  Note: This can only happen when the database for the MashZone NextGen Repository is hosted on a different server than the MashZone NextGen Server.

  Update the firewall configuration to allow this connection. Then restart the MashZone NextGen Server and confirm that this starts successfully.

- The URL or other connection configuration that you entered in MashZone NextGen for the MashZone NextGen Repository is incorrect.

  To correct an error in this case, edit the resource properties for the MashZone NextGen Repository in the *MashZoneNG-install* /apache-tomee-jaxrs/conf/tomee.xml file.

  Then restart the MashZone NextGen Server and confirm that this starts successfully.

- Port or connection configuration for the database is not set up properly to allow connections from the MashZone NextGen Server. See documentation for your database for more information.

Move MashZone NextGen and MashZone Repositories to Microsoft SQL Server

1. If you are using your LDAP Directory as the MashZone NextGen User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for MashZone NextGen *before* you move the MashZone NextGen Repository. See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

   When the MashZone NextGen User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the MashZone NextGen Repository or MashZone Repository in a new database, create the database following SQL Server documentation. Keep the following points in mind:

   - Make sure this database is supported by MashZone NextGen or MashZone. See "MashZone NextGen and MashZone Database Compatibility" on page 1567 for details.

     Note: The JTDS driver for SQL Server is the recommended driver for use with MashZone NextGen or MashZone and SQL Server databases due to known issues with the default SQL Server JDBC driver.

   - If you want MashZone NextGen to support international characters in meta-data for artifacts, make sure the database uses the UTF-16 character encoding and collation. See documentation for your database for specific instructions.

- It is a best practice to require passwords for every database account that can access the MashZone NextGen Repository.

3. Start the database(s) that will become host to the MashZone NextGen Repository and/or the MashZone Repository, if they are not already up.
4. Using the SQL tool for the database that will be host, add MashZone NextGen Repository tables with the scripts shown below from the corresponding folder in *MashZoneNG-install* /prestorepository/mssqldb:
   - createDBTables.txt for MetaData and the default User Repository
   - createSnapsTables.sql for Snapshots
   - createSchedulerTables.sql for Scheduler

   This folder also contains scripts to drop the corresponding MashZone NextGen Repository tables, if needed.
5. Copy the JAR file for the JDBC driver for your database to the following folder for each MashZone NextGen Server that uses this MashZone NextGen Repository or MashZone Server that uses this MashZone Repository:

   MashZoneNG-install /apache-tomee-jaxrs/lib
6. Replace the JAR for the MashZone NextGen Repository:
   a. Remove the *web-apps-home* /presto/WEB-INF/lib/jackbe-presto-rds-postgresql-derby.jar JAR file for each MashZone NextGen Server that uses this MashZone NextGen Repository. You can delete this JAR or simply move it to a folder that is not in the classpath for the application server that hosts MashZone NextGen.
   b. Copy this JAR file:

      MashZoneNG-install /prestorepository/jackbe-presto-rds-oracle-mysql-mssql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.
7. Update snapshot scheduler configuration for the MashZone NextGen Server:
   a. In the text editor of your choice, open the applicationContext-scheduler.xml file in the *webapps-home* /presto/WEB-INF/classes/ folder for the MashZone NextGen Server.
   b. Find the bean for `org.springframework.scheduling.quartz.SchedulerFactoryBean`.
   c. Update the `org.quartz.jobStore.driverDelegateClass` property to the `org.quartz.impl.jdbcjobstore.MSSQLDelegate` appropriate delegate for this database:
   d. Save this change.
   e. If this is a clustered environment, copy the updated applicationContext-scheduler.xml configuration file to each MashZone NextGen Server in the cluster.

8. Open the *MashZoneNG-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.
9. For the MashZone NextGen Repository, edit the <Resource> element with an ID of `MashzoneNextGenRepository` and:
   a. Update the JDBC driver, URL and credential properties:

```
<Resource id="MashzoneNextGenRepository" type="DataSource">
  JdbcDriver net.sourceforge.jtds.jdbc.Driver
  JdbcUrl jdbc:jtds:sqlserver://host-name:port/database
  UserName username
  Password password
  JtaManaged = false
</Resource>
```

   The JTA managed property *must* be `false`.

   b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

   Some common properties you may need to set include:

   - `validationQuery = select 1`
   - Common tuning properties for connections pools. See "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738.

10. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:
    a. Update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver net.sourceforge.jtds.jdbc.Driver
  JdbcUrl jdbc:jtds:sqlserver://host-name:port/database
  UserName username
  Password password
  defaultAutoCommit = true
  maxActive = 200
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
</Resource>
```

    b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

    Some common properties you may need to set include:

    - `validationQuery = select 1`
    - Common tuning properties for connections pools. See "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738.

11. Save your changes to this file.

    If the MashZone NextGen Server does not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.

12. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:
    a. Comment out the property definitions for the Derby database.
    b. Uncomment the properties for MSSQL and update these properties to match the JNDI configuration your previously updated for TomEE.
    c. Save your changes to this file.

    You must keep JDBC and JNDI configuration synchronized for the MashZone NextGen Repository. The application server is using JNDI to connect to the MashZone NextGen Repository, but some components still use JDBC information.
13. Restart the MashZone NextGen Server to apply these changes. This also restarts the MashZone Server.

    If the MashZone NextGen Server wedoes not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.
14. Update connection information for the Snapshots Repository:
    a. Open MashZone NextGen Hub and login.
    b. Add a JDBC driver for the new database that should host the Snapshots Repository. See "Add or Manage JDBC Drivers" on page 1681 for instructions on adding JDBC drivers.
    c. Expand the **JDBC Configuration** menu, if needed, and select **Datasources**.
    d. Select `SnapshotDatasource` and click **Edit**.
    e. Update configuration to point to the new database. See "Add a Data Source" on page 1679 for information on specific configuration properties.
    f. Click **Save**.
15. Restart the MashZone NextGen Server to apply these changes.
16. Load macros required for the Snapshot feature in MashZone NextGen:
    a. Open a command or terminal window and move to the *MashZoneNG-install* /presto-cli/bin folder.
    b. Enter the appropriate command, shown below, for your operating system:

For Windows	For Linux, OS/X or UNIX
`publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api`

Move the MashZone NextGen and MashZone Repositories to MySQL

1. If you are using your LDAP Directory as the MashZone NextGen User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for MashZone NextGen *before* you move the MashZone NextGen Repository. See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

   When the MashZone NextGen User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the MashZone NextGen Repository or MashZone Repository in a new database, create the database following MySQL documentation. Keep the following points in mind:

   - Make sure this database is supported by MashZone NextGen or MashZone. See "MashZone NextGen and MashZone Database Compatibility" on page 1567 for details.
   - If you want MashZone NextGen to support international characters in meta-data for artifacts, set the character encoding and collation to UTF-8 when you create the database. See documentation for your database for specific instructions.
   - For medium or larger MySQL databases that will host the MashZone NextGen Repository, you should increase the maximum allowed packet size, which defaults to 1MB, for the database.

3. Start the database(s) that will become host to the MashZone NextGen Repository and/or the MashZone Repository, if they are not already up.

4. Using the SQL tool for the database that will be host, add MashZone NextGen Repository tables with the scripts shown below from the corresponding folder in *MashZoneNG-install* /prestorepository/mysql:

   a. Create a Database to hold the MashZone NextGen Repository tables. See the file createDB.txt for an example.

   b. Create a User with rights to the database created in step **a**. See the file createUser.txt for an example.

   c. Connect to the MySQL database created in step **a** using a SQL tool (for example MySQL command line client) using the user created in step **b**. See the comments in the file createDBTables.txt for examples.

   d. Execute the statements in the file createDBTables.txt to create the tables using the SQL tool (for example, use the MySQL `source` command: `"source /path/to/createDBTables.txt"`).

   e. Execute the statements in the file createSchedulerTables.sql (for example: `"source /path/to/createSchedulerTables.sql"`).

   f. Execute the statements in the file createSnapsTables.sql (for example: `"source /path/to/createSnapsTables.sql"`).

This folder also contains scripts to drop the corresponding MashZone NextGen Repository tables, if needed.

5. Replace the JAR for the MashZone NextGen Repository:
   a. Remove the *MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/lib/jackbe-presto-rds-postgresql-derby.jar JAR file for each MashZone NextGen Server that uses this MashZone NextGen Repository. You can delete this JAR or simply move it to a folder that is not in the classpath for the application server that hosts MashZone NextGen.
   b. Copy this JAR file:

      MashZoneNG-install /prestorepository/jackbe-presto-rds-oracle-mysql-mssql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.
6. Copy the MySQL JDBC driver jar file to *MashZoneNG-install* /apache-tomee-jaxrs/lib.
7. Open the *MashZoneNG-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.
8. For the MashZone NextGen Repository, edit the <Resource> element with an ID of `MashzoneNextGenRepository` and:
   a. Update the JDBC driver, URL and credential properties:

```
<Resource id="MashzoneNextGenRepository" type="DataSource">
  JdbcDriver com.mysql.jdbc.Driver
  JdbcUrl jdbc:mysql://host-name/databasename
  UserName = username
  Password = password
  JtaManaged = false
</Resource>
```

      For MySQL databases, it is *recommended* that you include the database name in data source URLs. If this information is omitted, testing the data source fails and may also cause errors with access to stored procedures.

      The JTA managed property *must* be `false`.
   b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

      Some common properties you may need to set include:

      - `validationQuery = select 1 from dual`
      - Common tuning properties for connections pools. See "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738.
9. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:
   a. Update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver com.mysql.jdbc.Driver
  JdbcUrl jdbc:mysql://host-name:port/databasename
  UserName = username
```

```
  Password = password
  defaultAutoCommit = true
  maxActive = 200
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
</Resource>
```

For MySQL databases, it is *recommended* that you include the database name in data source URLs. If this information is omitted, testing the data source fails and may also cause errors with access to stored procedures.

b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

Some common properties you may need to set include:

- `validationQuery = select 1`
- Common tuning properties for connections pools. See "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738.

10. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

a. Comment out the property definitions for the Derby database.

b. Uncomment the properties for MySQL and update these properties to match the JNDI configuration your previously updated for TomEE.

c. Save your changes to this file.

You must keep JDBC and JNDI configuration synchronized for the MashZone NextGen Repository. The application server is using JNDI to connect to the MashZone NextGen Repository, but some components still use JDBC information.

11. Start the MashZone NextGen Server to apply these changes. This also starts the MashZone Server.

If the MashZone NextGen Server wedoes not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.

12. Update connection information for the Snapshots Repository:

a. Open MashZone NextGen Hub and login.

b. Add a JDBC driver for the new database that should host the Snapshots Repository. See "Add or Manage JDBC Drivers" on page 1681 for instructions on adding JDBC drivers.

c. Expand the **JDBC Configuration** menu, if needed, and select **Datasources**.

d. Select `SnapshotDatasource` and click **Edit**.

e. Update configuration to point to the new database. See "Add a Data Source" on page 1679 for information on specific configuration properties.

f. Click **Save**.

13. Restart the MashZone NextGen Server to apply these changes.
14. Load macros required for the Snapshot feature in MashZone NextGen:
    a. Open a command or terminal window and move to the *MashZoneNG-install* /presto-cli/bin folder.
    b. Enter the appropriate command, shown below, for your operating system:

For Windows	For Linux, OS/X or UNIX
`publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api`

Move the MashZone NextGen and MashZone Repositories to Oracle

1. If you are using your LDAP Directory as the MashZone NextGen User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for MashZone NextGen *before* you move the MashZone NextGen Repository. See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

   When the MashZone NextGen User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the MashZone NextGen Repository or MashZone Repository in a new database, create the database following Oracle documentation.
   - Make sure this database is supported by MashZone NextGen or MashZone. See "MashZone NextGen and MashZone Database Compatibility" on page 1567 for details.
   - If you want MashZone NextGen to support international characters in meta-data for artifacts, set the character encoding to AL32UTF8 when you create the database. See documentation for your database for specific instructions.
   - It is a best practice to require passwords for every database account that can access the MashZone NextGen Repository.
3. Start the database(s) that will become host to the MashZone NextGen Repository and/or the MashZone Repository, if they are not already up.
4. Using the SQL tool for the database that will be host, add MashZone NextGen Repository tables with the scripts shown below from the corresponding folder in *MashZoneNG-install* /prestorepository/oracledb:
   - createDBTables.txt for MetaData and the default User Repository
   - createSnapsTables.sql for Snapshots

- createSchedulerTables.sql for Scheduler

Note: This folder contains other scripts to drop the corresponding MashZone NextGen Repository tables.

5. Replace the JAR for the MashZone NextGen Repository:
   a. Remove the *web-apps-home* /presto/WEB-INF/lib/jackbe-presto-rds-postgresql-derby.jar JAR file for each MashZone NextGen Server that uses this MashZone NextGen Repository. You can delete this JAR or simply move it to a folder that is not in the classpath for the application server that hosts MashZone NextGen.
   b. Copy this JAR file:

      MashZoneNG-install /prestorepository/jackbe-presto-rds-oracle-mysql-mssql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.
6. Copy the JAR file for the JDBC driver for your database to the following folder for each MashZone NextGen Server that uses this MashZone NextGen Repository or MashZone Server that uses this MashZone Repository:

   MashZoneNG-install /apache-tomee-jaxrs/lib
7. Add the Quartz-Oracle JAR to the MashZone NextGen Server:
   a. Download quartz-oracle-1.7.2.jar.
      a. In a web browser open the link: .
      b. Click the **Download (Jar)** link to save the file.
   b. Copy the quartz-oracle-1.7.2.jar file to the *MashZoneNG-install* /apache-tomee-jaxrs/lib folder.

      If this is a clustered environment, repeat step b for each member of the cluster.
8. Update snapshot scheduler configuration for the MashZone NextGen Server:
   a. In the text editor of your choice, open the applicationContext-scheduler.xml file in the *webapps-home* /presto/WEB-INF/classes/ folder for the MashZone NextGen Server.
   b. Find the `org.springframework.scheduling.quartz.SchedulerFactoryBean` bean.
   c. Update the `org.quartz.jobStore.driverDelegateClass` property to the `org.quartz.impl.jdbcjobstore.oracle.OracleDelegate` delegate.

      The configuration would now look like:

```
...
<bean id="scheduler">
  <property name="applicationContextSchedulerContextKey">
    <value>applicationContext</value>
  </property>
  <property name="quartzProperties">
    <props>
      <prop key="org.quartz.scheduler.instanceId">AUTO</prop>
```

```
      <prop key="org.quartz.jobStore.class"> org.quartz.impl.jdbcjobstore.
                JobStoreTX</prop>
      <prop key="org.quartz.jobStore.tablePrefix">QRTZ_</prop>
      <prop key="org.quartz.jobStore.driverDelegateClass"> org.quartz.impl.
                jdbcjobstore.oracle.OracleDelegate</prop>
      <prop key="org.quartz.jobStore.dataSource">schedulerDS</prop>
      ...
    </props>
  </property>
</bean>
...
```

d. Save this change.

e. If this is a clustered environment, copy the updated applicationContext-scheduler.xml configuration file to each MashZone NextGen Server in the cluster.

9. Open the *MashZoneNG-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.

10. For the MashZone NextGen Repository, edit the <Resource> element with an ID of `MashzoneNextGenRepository` and:

    a. Update the JDBC driver, URL and credential properties:

```
<Resource id="MashzoneNextGenRepository" type="DataSource">
  JdbcDriver oracle.jdbc.OracleDriver
  JdbcUrl jdbc:oracle:drivertype:@host-name:port:dbname
  UserName username
  Password password
  JtaManaged false
</Resource>
```

    The JTA managed property *must* be `false`.

    b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

    Some common properties you may need to set include:

    - `validationQuery = select 1 from dual`
    - Common tuning properties for connections pools. See "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738.

11. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:

    a. For Oracle 11g, update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver oracle.jdbc.OracleDriver
  JdbcUrl jdbc:oracle:thin//:host-name:port/sid
  UserName username
  Password password
  defaultAutoCommit = true
  rollbackOnReturn = true
  removeAbandoned = true
  logAbandoned = true
  removedAbandonedTimeout = 5
  maxActive = 200
  maxIdle = 20
  maxOpenPreparedStatements = 0
```

```
  maxWaitTime = 5000
</Resource>
```

b. For Oracle 12c, update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver oracle.jdbc.OracleDriver
  #Oracle 12 url
  JdbcUrl jdbc:oracle:drivertype:@host-name:port:dbname
  UserName username
  Password password
  defaultAutoCommit = true
  rollbackOnReturn = true
  removeAbandoned = true
  logAbandoned = true
  removedAbandonedTimeout = 5
  maxActive = 100
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
</Resource>
```

12. Save your changes to this file.

    If the MashZone NextGen Server does not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.

13. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

    a. Comment out the property definitions for the Derby database.

    b. Uncomment the properties for Oracle and update these properties to match the JNDI configuration your previously updated for TomEE.

    c. Save your changes to this file.

    You must keep JDBC and JNDI configuration synchronized for the MashZone NextGen Repository. The application server is using JNDI to connect to the MashZone NextGen Repository, but some components still use JDBC information.

14. Restart the MashZone NextGen Server to apply these changes. This also restarts the MashZone Server.

    If the MashZone NextGen Server wedoes not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.

15. Update connection information for the Snapshots Repository:

    a. Open MashZone NextGen Hub and login.

    b. Add a JDBC driver for the new database that should host the Snapshots Repository. See "Add or Manage JDBC Drivers" on page 1681 for instructions on adding JDBC drivers.

    c. Expand the **JDBC Configuration** menu, if needed, and select **Datasources**.

    d. Select `SnapshotDatasource` and click **Edit**.

e. Update configuration to point to the new database. See "Add a Data Source" on page 1679 for information on specific configuration properties.

f. Click **Save**.

16. Restart the MashZone NextGen Server to apply these changes.

17. Load macros required for the Snapshot feature in MashZone NextGen:

a. Open a command or terminal window and move to the *MashZoneNG-install* / presto-cli/bin folder.

b. Enter the appropriate command, shown below, for your operating system:

For Windows	For Linux, OS/X or UNIX
`publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api`

Move the MashZone NextGen and MashZone Repositories to PostGres

1. If you are using your LDAP Directory as the MashZone NextGen User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for MashZone NextGen *before* you move the MashZone NextGen Repository. See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

   When the MashZone NextGen User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the MashZone NextGen Repository or MashZone Repository in a new database, create the database following PostGreSQL documentation. Keep the following points in mind:

   - Make sure this database is supported by MashZone NextGen or MashZone. See "MashZone NextGen and MashZone Database Compatibility" on page 1567 for details.
   - If you want MashZone NextGen to support international characters in meta-data for artifacts, set the character encoding to UTF8 when you create the database. See documentation for your database for specific instructions.
   - It is a best practice to require passwords for every database account that can access the MashZone NextGen Repository.
   - When you initialize the PostGres database that will host the MashZone NextGen Repository or MashZone Repository, you may need to specifically set the locale used by the database to ensure case-insensitive sorting.

3. Start the database(s) that will become host to the MashZone NextGen Repository and/or the MashZone Repository, if they are not already up.
4. Using the SQL tool for the database that will be host, add MashZone NextGen Repository tables with the scripts shown below from the corresponding folder in *MashZoneNG-install* /prestorepository/postgresdb:
   - createDBTables.txt for MetaData and the default User Repository
   - createSnapsTables.sql for Snapshots
   - createSchedulerTables.sql for Scheduler

   There are also scripts to drop the corresponding MashZone NextGen Repository tables in these folders, if needed.
5. Update snapshot scheduler configuration for the MashZone NextGen Server:
   a. In the text editor of your choice, open the applicationContext-scheduler.xml file in the *webapps-home* /presto/WEB-INF/classes/ folder for the MashZone NextGen Server.
   b. Find the `org.springframework.scheduling.quartz.SchedulerFactoryBean` bean.
   c. Update the `org.quartz.jobStore.driverDelegateClass` property to the `org.quartz.impl.jdbcjobstore.PostgreSQLDelegate` delegate.

      The configuration would now look like:

```
...
<bean id="scheduler"
>
  <property name="applicationContextSchedulerContextKey">
    <value>applicationContext</value>
  </property>
  <property name="quartzProperties">
    <props>
      <prop key="org.quartz.scheduler.instanceId">AUTO</prop>
      <prop key="org.quartz.jobStore.class"> org.quartz.impl.jdbcjobstore.JobStoreTX</prop
      <prop key="org.quartz.jobStore.tablePrefix">QRTZ_</prop>
      <prop key="org.quartz.jobStore.driverDelegateClass"> org.quartz.impl.jdbcjobstore.Po
      <prop key="org.quartz.jobStore.dataSource">schedulerDS</prop>
      ...
    </props>
  </property>
</bean>
...
```

   d. Save this change.
   e. If this is a clustered environment, copy the updated applicationContext-scheduler.xml configuration file to each MashZone NextGen Server in the cluster.
6. Copy the JAR file for the JDBC driver for your database to the following folder for each MashZone NextGen Server that uses this MashZone NextGen Repository or MashZone Server that uses this MashZone Repository: *MashZoneNG-install* /apache-tomee-jaxrs/lib.

7. Open rdsApplicationContext.xml under *MashZoneNG-install* /apache-tomee-jaxrs/classes and add the following keys to:

```
<property name="jdoProperties">
...
<map>
...
<entry key="javax.jdo.mapping.Schema" value="public"/>
<entry key="datanucleus.identifier.case" value="LowerCase"/>
...
</map>
</property>
```

8. If you are using PostgreSQL version 9.x please open postgresql.conf under PostgreSQL-install/9.x/data and un-comment the following property and make sure it is set to **off**: `standard_conforming_strings = off`.

9. Open the *MashZoneNG-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.

10. For the MashZone NextGen Repository, edit the <Resource> element with an ID of `MashzoneNextGenRepository` and:

    a. Update the JDBC driver, URL and credential properties:

```
<Resource id="MashzoneNextGenRepository" type="DataSource">
  JdbcDriver org.Postgresql.Driver
  JdbcUrl jdbc:postgresql://host-name:port/databasename
  UserName username
  Password password
  JtaManaged = false
</Resource>
```

    The JTA managed property *must* be `false`.

    b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

    Some common properties you may need to set include:

    - `validationQuery = select 1`
    - Common tuning properties for connections pools. See "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738.

11. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:

    a. Update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver org.Postgresql.Driver
  JdbcUrl jdbc:postgresql//:host-name:port/databasename
  UserName username
  Password password
  defaultAutoCommit = true
  maxActive = 200
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
</Resource>
```

12. Save your changes to this file.

    If the MashZone NextGen Server does not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.

13. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

    a. Comment out the property definitions for the Derby database.

    b. Uncomment the properties for PostGres and update these properties to match the JNDI configuration your previously updated for TomEE.

    c. Save your changes to this file.

    You must keep JDBC and JNDI configuration synchronized for the MashZone NextGen Repository. The application server is using JNDI to connect to the MashZone NextGen Repository, but some components still use JDBC information.

14. Restart the MashZone NextGen Server to apply these changes. This also restarts the MashZone Server.

    If the MashZone NextGen Server wedoes not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 1574 for suggestions.

15. Update connection information for the Snapshots Repository:

    a. Open MashZone NextGen Hub and login.

    b. Add a JDBC driver for the new database that should host the Snapshots Repository. See "Add or Manage JDBC Drivers" on page 1681 for instructions on adding JDBC drivers.

    c. Expand the **JDBC Configuration** menu, if needed, and select **Datasources**.

    d. Select `SnapshotDatasource` and click **Edit**.

    e. Update configuration to point to the new database. See "Add a Data Source" on page 1679 for information on specific configuration properties.

    f. Click **Save**.

16. Restart the MashZone NextGen Server to apply these changes.

17. Load macros required for the Snapshot feature in MashZone NextGen:

    a. Open a command or terminal window and move to the *MashZoneNG-install* /presto-cli/bin folder.

    b. Enter the appropriate command, shown below, for your operating system:

For Windows	For Linux, OS/X or UNIX
`publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api`

Integrate Your LDAP Directory with MashZone NextGen

In many cases, users and authentication information for an organization is defined in an existing LDAP Directory. You can configure MashZone NextGen to use your LDAP Directory as the source for user and group information.

Note: See the *System Requirements for Software AG Products* guide for information on MashZone NextGen support for specific LDAP Directory solutions.

To configure your LDAP Directory as the MashZone NextGen User Repository:

1. If the MashZone NextGen Server is not yet started, start MashZone NextGen. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Change MashZone NextGen configuration to use LDAP as the authentication provider.
   a. Edit the userRepositoryApplicationContext.xml file in the *MashZoneNG-config* folder with any text editor.

      Note: This folder may be in the default location or in an external location. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

   b. Remove the comment markers around this statement: `<import resource="/userRepositoryApplicationContext-ldap.xml">`.
   c. Comment out this statement: `<import resource="/userRepositoryApplicationContext-jdbc.xml">` property.

      Note: You cannot use both default authentication and LDAP authentication.

   The configuration should look something like this:

```
<beans>
  <!-- Choose between the interal JDBC repository and LDAP comment/uncomment
    these two import statements -->
  <import resource="/userRepositoryApplicationContext-ldap.xml">
  <!-- <import resource="/userRepositoryApplicationContext-jdbc.xml"> -->
...
</beans>
```

3. Change MashZone NextGen configuration for the user attribute provider.

a. If it is not already open, edit the userRepositoryApplicationContext.xml file in the *MashZoneNG-config* folder with any text editor.

> **Note:** This folder may be in the default location or in an external location. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

b. Find the `userAttributeProvider` bean:

```
<bean id="userAttributeProvider"
>
...
```

c. Remove comment markers around the `ldapAttributeProvider` bean reference in the providers property list.

The configuration should now look something like this:

```
<bean id="userAttributeProvider"
>
  <property name="providers">
    <list>
      <ref bean="ldapAttributeProvider"/>
      <ref bean="internalUserAttributeProvider"/>
    </list>
  </property>
</bean>
```

d. Save your changes to this file.

> **Important:** Do *not* restart the MashZone NextGen Server until all other LDAP configuration steps have been completed.

4. Define configuration in the Admin Console in MashZone NextGen Hub for:

- Connections to the LDAP Directory. See "Defining LDAP Connection Configuration" on page 1592.
- Authentication mechanisms. See "Defining the Authentication Scheme" on page 1592.
- Authorization mechanisms. See "Defining the Authorization Scheme" on page 1593.
- All user and group queries used in MashZone NextGen applications. See "Enabling MashZone NextGen Application Queries for All LDAP Users or Groups for Permissions" on page 1595.

See also "Expose User Attributes from the User Repository in MashZone NextGen" on page 1689 for information on making LDAP user attributes accessible as MashZone NextGen user attributes.

5. Add the built-in MashZone NextGen user groups to LDAP as new groups and assign at least one user as a MashZone NextGen administrator.

See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

6. Restart the MashZone NextGen Server.

   MashZone NextGen now uses LDAP as the user repository. You can now login using the user account assigned in earlier steps as a MashZone NextGen administrator.

   To grant access to other users, add them to an appropriate built-in MashZone NextGen user group in LDAP. See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

Defining LDAP Connection Configuration

1. If needed, log into MashZone NextGen Hub and click Admin Console in the main menu.
2. Expand **MashZone NextGen Repositories** and click **User Repository - LDAP**.
3. Set these properties for your LDAP Directory:
   - *LDAP URL* = the URL to your LDAP directory. For example:

     ```
     ldap://localhost:389/dc=somecompany,dc=com
     ```

   - *Directory User Name* = the distinguished name for the user account to connect to this LDAP Directory. This *must* be a privileged account. For example:

     ```
     uid=admin,ou=system
     ```

   - *Directory Password* = the password for the user to connect to this LDAP Directory.
4. Change any advanced options (see "Defining the Authentication Scheme" on page 1592, "Defining the Authorization Scheme" on page 1593 and "Enabling MashZone NextGen Application Queries for All LDAP Users or Groups for Permissions" on page 1595) and save your changes.

Defining the Authentication Scheme

Authentication against LDAP determines if a distinguished name exists for a user. This searches for a user entry based on a specific username. Search-based authentication works, for example, if user names are users' email addresses.

To define the authentication scheme:

1. If needed, log into MashZone NextGen Hub and click Admin Console in the main menu.
2. Expand **MashZone NextGen Repositories** and click **User Repository - LDAP**.
3. Click **Advanced Options**.
4. Set these properties in the Authentication Properties section:
   - *User Search Base* = the base context for a user search in authentication. This produces a list of all users which is filtered with a combination of the User Search Filter and User Search Subtree properties to authenticate a user. For example:

     ```
     ou=users,ou=system
     ```

- *User Search Filter* = the relative filter to apply to search for users during authentication. The variable `{0}` is replaced with the user's username from login.

  This filter is based from the context defined in User Search Base. For example:

  ```
  email={0}
  ```

  Note: This attribute *must* be the same attribute used in the **User ID Attribute Name** property.

- *User Search Subtree* = set this option if the search should be recursive through all levels of the Directory under the search base. If you clear this option, search only checks direct children of the search base.
- **Use LDAP VLV Control for Sorting and Paging** = this option is set by default to allow MashZone NextGen to use *virtual list views* (VLV) to paginate and sort LDAP search results.

  Most LDAP directories support VLV, so in most cases you can leave this option set. If your LDAP directory logs errors for "unsupported search control", you can use this option to turn VLV off.

- *User ID Attribute Name* = the LDAP attribute that contains the username that users login with. For example:

  ```
  email
  ```

  This value becomes the user ID for all further security contexts, unless the User ID Pattern property is also set.

- *User ID Pattern* = a regular expression that is applied to user login names to extract the user ID for all further security contexts. This is only applied after authentication occurs.

Defining the Authorization Scheme

MashZone NextGen permissions are assigned to user groups or to individual users. To set up authorization when LDAP is the user repository, you must relate MashZone NextGen user groups to user groups in LDAP and define how users are assigned to groups in LDAP. User membership in LDAP groups can be defined by adding users to group entries or by adding group names to user entries, but *not* both.

Note: In previous releases, MashZone NextGen user groups were called roles that could be implemented as user roles in LDAP instead of user groups. To use roles in LDAP for authorization in MashZone NextGen, please contact your Software AG representative for more information.

You *must* add the built-in MashZone NextGen groups that define basic permissions as groups in LDAP. You assign users to these built-in groups to assign basic MashZone NextGen permissions. Your existing LDAP groups can then be used in MashZone NextGen to define run permissions for specific mashables, mashups or apps. For more

information on authorization, see "Authorization Policies and Permissions" on page 1619.

1. If needed, log into MashZone NextGen Hub and click Admin Console in the main menu.
2. Expand **MashZone NextGen Repositories** and click **User Repository - LDAP**.
3. Click **Advanced Options.**
4. If user membership is defined in group entries in your LDAP directory, set these properties:
   - Set the **Search Groups for User Membership** option.
   - Enter the beginning context for user group searches in the *Group Search Base* property.

     This is combined with the User Group Search Filter to find LDAP groups to determine user membership in groups that may have MashZone NextGen permissions. For example:

     ```
     ou=groups
     ```

   - Enter the filter to apply in group searches in the *User Group Search Filter* property.

     This is combined with Group Search Base to find LDAP groups to determine user membership in groups that may have MashZone NextGen permissions. The variable `{0}` is replaced with the user's username from login. For example:

     ```
     uniquemember={0}
     ```

   - Enter the LDAP attribute in group entries that identifies a group in the *Group Name Attribute* property.

     This attribute contains the name of user groups that is used in MashZone NextGen permissions. The default value is the group common name:

     ```
     cn
     ```

     Important: If you change this property, you *must* also update the **Group Name Pattern** property.

   - If group IDs in your LDAP Directory are not simple common names (see Group Name Attribute), enter a regular expression in **Group Name Pattern** to identify the built-in MashZone NextGen groups.

     For example:

     ```
     cn=(PRESTO_.*?)
     ```

     MashZone NextGen expects specific names for the built-in groups that you add to your LDAP Directory. These values are defined in the common name of the group. This property allows MashZone NextGen to find the expected values for built-in groups, but use the full correct group names for the groups for your organization.

5. If user membership is defined *solely* in user entries, set these properties:
   - Clear the **Search Groups for User Membership** option.
   - Enter the name of the LDAP attribute in user entries that identies the groups that users belong to in the **User Membership Attribute** property.
   - If group IDs in your LDAP Directory are not simple common names, enter a regular expression in **Group Name Pattern** to identify the built-in MashZone NextGen groups.

     For example:

     ```
     cn=(PRESTO_.*?)
     ```

     MashZone NextGen expects specific names for the built-in groups that you add to your LDAP Directory. These values are defined in the common name of the group. This property allows MashZone NextGen to find the expected values for built-in groups, but use the full correct group names for the groups for your organization.

With these properties set, for example:

```
Search Groups for User Membership = true
Group Search Base=ou=groups
User Group Search Filter=uniquemember={0}
Group Name Attribute = cn
```

And a username of `jwalker`, MashZone NextGen would search all entries in `ou=groups` where `uniquemember=jwalker`. The names for any of these groups would be the common name (cn) for the group entry.

If these properties were set instead:

```
Search Groups for User Membership = false
User Membership Attribute = memberOf
```

The list of groups would consist of all values in the `memberOf` attribute in the `jwalker` user entry.

This list of group names would be compared to the built-in MashZone NextGen groups and to groups with run permissions for artifacts to determine the full set of permissions for `jwalker`.

Enabling MashZone NextGen Application Queries for All LDAP Users or Groups for Permissions

MashZone NextGen queries the User Repository for user groups and users to enable you and other users to assign permissions for MashZone NextGen resources. To enable these queries you set properties in the Admin Console:

1. If needed, log into MashZone NextGen Hub and click Admin Console in the main menu.
2. Expand **MashZone NextGen Repositories** and click **User Repository - LDAP**.
3. Click **Advanced Options**.

4. To enable queries for all users, set these properties:

   - *User Search Base* (in Authentication Properties) = the base context for a search for all users. This is used with the All Users Search Filter and Search Subtree For All Users properties to get a result. For example:

     ```
     ou=People
     ```

     Important: This property is also used to search for users during authentication. Consider both uses before changing its value.

   - *All Users Search Filter* (in MashZone NextGen Queries) = the search filter, combined with User Search Base that is used to find all user entries. For example:

     ```
     objectclass=inetOrgPerson
     ```

     Ensure that the `objectclass=inetOrgPerson` attribute is set on the LDAP server.

   To support wildcard searches and define the sort order for results, you must also define these properties:

   - **Attributes Used in Wildcard Search** (in MashZone NextGen Queries) = a list of LDAP attributes, separated by commas, to search in for wildcard searches. This defaults to:

     ```
     cn,uid
     ```

   - **User Sort By Attribute** (in MashZone NextGen Queries) = the LDAP attribute that should be used to sort the results of wildcard searches. This defaults to:

     ```
     cn
     ```

   You must also define these properties so that Admin Console can display minimal user information:

   - *User First Name Attribute* (in MashZone NextGen Queries) = the LDAP attribute that holds users' first names.
   - *User Last Name Attribute* (in MashZone NextGen Queries) = the LDAP attribute that holds users' last names.
   - *User Email Attribute* (in MashZone NextGen Queries) = the LDAP attribute that holds users' email addresses.

5. To enable queries for LDAP groups that can be used to assign MashZone NextGen permissions:

   - *Group Search Base* (in Authorization Properties) = the beginning context, combined with Filter to Find All Groups for Roles to find all LDAP groups that can be used to assign MashZone NextGen permissions. For example:

     ```
     ou=groups
     ```

Important: This property is also used to search for MashZone NextGen permissions during authorization. Consider both uses before changing its value.

- *Filter to Find All Groups for Permissions* = the search filter, combined with Group Search Base that is used to find all LDAP groups that may be used to assign MashZone NextGen permissions. For example:

  ```
  objectclass=groupOfUniqueNames
  ```

Trouble shooting: If your LDAP user with role MashZone NextGen_Administrator does not work, it might be helpful to stop MashZone NextGen first, deactivate and reactivate your LDAP connection in MashZone NextGen and then restart MashZone NextGen again.

Use the Default MashZone NextGen User Repository

MashZone NextGen authenticates users against a repository and retrieve authorization from the repository. This can be either a database or an LDAP Directory.

MashZone NextGen is installed with a default user repository within the MashZone NextGen Repository. You can use this initially as you explore MashZone NextGen or keep as the permanent source for user information, if desired. The default user repository also contains Default User Accounts that you can use initially.

Important: TheMashZone NextGenRepository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

No configuration is required to use the default user repository. If you use this default, you manage user and group information using functions in the Admin Console. See these topics for more information:

- "Manage Users" on page 1597
- "Manage User Groups" on page 1599
- "Automatically Assign New Users to Groups" on page 1600

Manage Users

If you are using the default MashZone NextGen User Repository, MashZone NextGen administrators can add users, assign them to groups to grant them permissions for various actions, and otherwise manage users in the Admin Console to. See "Create Users" on page 1598 and "Edit, Grant Permissions and other User Management Tasks" on page 1598 for instructions.

If you have configured MashZone NextGen to use your LDAP Directory as the User Repository, you manage users and groups and assign permissions in LDAP.

Create Users

1. If needed, open the Admin Console (click in the MashZone NextGen Hub main menu).
2. Expand the **Users and Groups** tab and click **Users**.
3. Click **Add new user** and complete the following information:
   - **User Name** must be unique. This is the end-user's login name and primary key for the user record. User names cannot exceed 256 characters.
   - **First Name** and **Last Name** are optional.
   - **Email**
   - **Password** cannot exceed 50 characters. Confirm the password.

   Note: Valid characters for the User Name and Password fields are defined by the database you use for the default MashZone NextGen User Repository.
4. Set the **Active** option, if needed, and click **Add User**.
5. Add another user or click **Done** to close this form.

The new user is now active in MashZone NextGen and has permissions as an authenticated user to work in the AppDepot. They can also run any artifact that grants run permissions to the All Authenticated Users group.

To give new users access to the MashZone NextGen Hub to create artifacts or simply to work with artifacts from other users, you must add them to other groups, either manually or automatically. For more information, see:

- "Automatically Assign New Users to Groups" on page 1600
- "Edit, Grant Permissions and other User Management Tasks" on page 1598

Edit, Grant Permissions and other User Management Tasks

1. If needed, open the Admin Console (click in the MashZone NextGen Hub main menu).
2. Expand the **Users and Groups** tab and click **Users**.
3. Find the user in the list using Search or scrolling. Management options include:
   - Click r a new password.
   - Click er's name, login name or email.
   - Click **Change user status** to activate the user or click **Change user status** to deactivate.
   - Click **Manage User Groups** to grant or manage permissions for this user.

- Enter part of a group name, if needed, and click **Search** to see a list of user groups with permissions and the user groups/permissions currently assigned to this user.
- Click a group from the left column to assign a user group/permission to this user. Click a group from the right pane to remove a user group/permission.

  See "Built-In MashZone NextGen User Groups and Permissions" on page 1622 for information on the permissions for the built-in user groups in MashZone NextGen. All other user groups grant run permissions to specific mashables, mashups or apps.
- Click **Save changes** to update groups and permissions for this user.

- Click **Delete** and confirm. Note that you cannot delete the default administrator account (the `admin` user) for MashZone NextGen.

Manage User Groups

Groups are used to grant permissions to users for specific actions in MashZone NextGen. If you are using the default MashZone NextGen User Repository, MashZone NextGen administrators add and manage user groups in the Admin Console. If you have configured MashZone NextGen to use your LDAP Directory as the User Repository, you define and manage user groups in your LDAP Directory.

User groups contain of a set of users. Groups also have one or more permissions assigned.

Built-in MashZone NextGengroups have a set of permissions predefined that allow users to work in the MashZone NextGen Hub and the AppDepot with *one exception*: the permission to run mashable information sources, mashups and apps is assigned by *user-defined groups*. See "Built-In MashZone NextGen User Groups and Permissions" on page 1622 for details.

Note: MashZone NextGen administrators and artifact owners automatically have permission to run all artifacts or the artifacts they own repsectively.

User-defined groups are groups that administrators add to MashZone NextGen. These groups are used to grant run permissions to group members for the specific artifacts that the group has been added to. MashZone NextGen administrators can also automatically grant run permissions to groups to run all apps, all mashups, all mashables or any combination. See "Automatically Grant Run Permissions to Users and Groups" on page 1624 for information.

1. If needed, open the Admin Console (click in the MashZone NextGen Hub main menu).
2. Expand the **Users and Groups** tab and click **Groups**.

   A list of groups displays. You can filter this list by entering part of a group name and clicking **Search**.

3. Click **Add new user group** and:
   a. Enter a unique name in the **Group** field.
   b. Click **Add Group**.
   c. Add more groups or click **Done** to close this form.
4. To delete a user group, click **Delete** and confirm this.

Important: Do *not* delete any of the built-in groups for MashZone NextGen: `MashZone NextGen_Administrator`, `MashZone NextGen_Developer`, `MashZone NextGen_PowerUser`, `MashZone NextGen_AuthenticatedUser` or `MashZone NextGen_Guest`.

Automatically Assign New Users to Groups

If you are using the Default MashZone NextGen User Repository, you can automatically assign new users to groups when you add them to MashZone NextGen. This can simplify some of the process of granting permissions to users.

Note: The feature is not available if you are using your LDAP Directory as the MashZone NextGen User Repository.

You can add both built-in MashZone NextGen groups or groups that you have added for your organization to the list of default groups to automatically assign to new users. If you make `Presto_PowerUser` a default group, for example, every new user would be granted access to MashZone NextGen Hub with permission to register mashables, create mashups in Wires, create basic apps and use Mashboard to create workspace apps.

If you also added a group named `runRss`, assigned this group in run permissions for every mashable that is a web feed and then added it to the default groups, all new users would automatically be granted run permissions for every existing web feed in MashZone NextGen.

To assign new users to groups

1. If needed, open the Admin Console (click in the MashZone NextGen Hub main menu).
2. Expand the **Users and Groups** tab and click **Default Groups**.
3. Enter part of a group name and click **Search**.

   A list of groups that start with that name displays.
4. Drag a group and drop it in the **Default User Groups** bucket.

   Continue to find and add groups as needed.
5. Click **Save these changes**.

New user will automatically be assigned to all groups in the default list. This change does *not* affect any existing users.

What's Next

When MashZone NextGen is installed as an integral part of IBO, there are several optional tasks you may need to complete after the initial installation to support business intelligence dashboards for users. This includes:

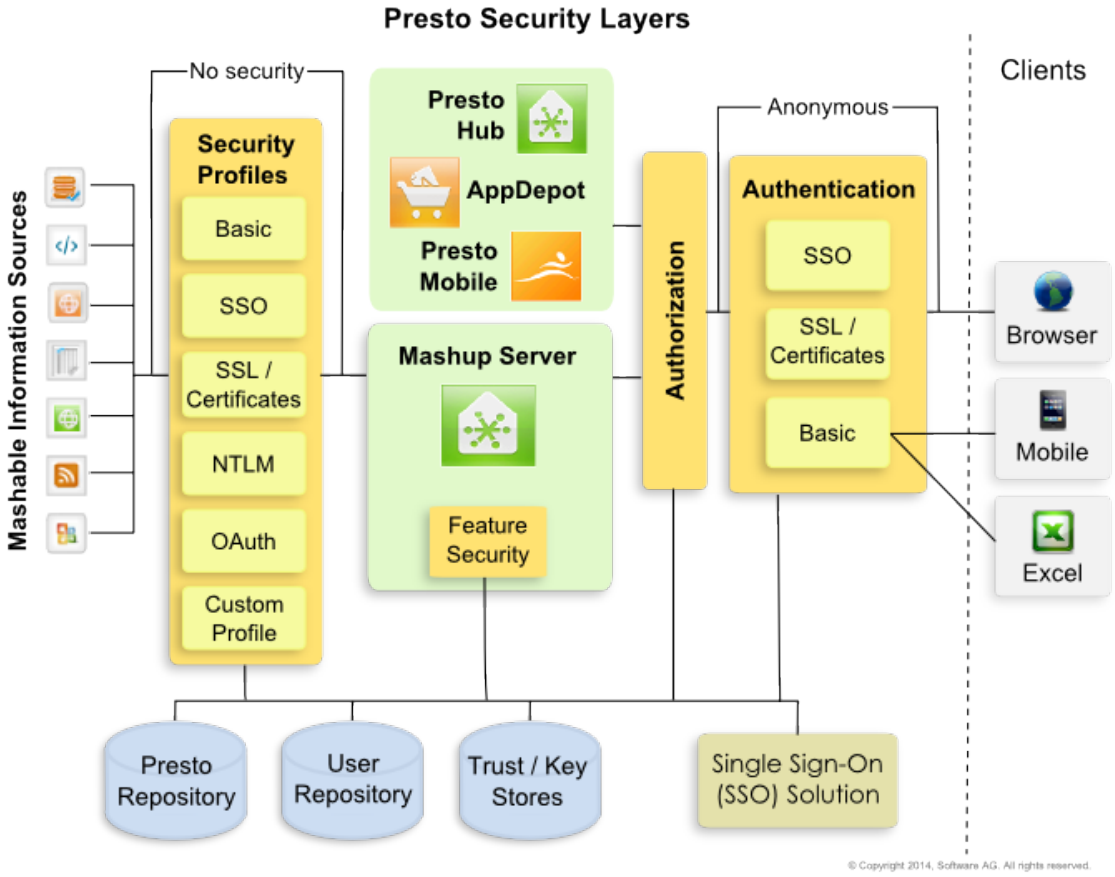
- Configuration for Process Performance Manager (PPM) Integration to allow users to use PPM charts or data in workspace apps.
- Configuration for MashZone NextGen connections and subscriptions to business process events from the EDA. See "Event Service Configuration and Administration" on page 1704 for links to instructions.
- Configuration for MashZone NextGen connections and subscriptions to Apama events. See "Event Service Configuration and Administration" on page 1704 for links to instructions.
- Configuration to enable business process dashboards. See the *Working with Business Process Dashboards* guide.

Other common configuration and administration tasks for MashZone NextGen include:

- The full set of MashZone NextGen Server Configuration includes:
  - Common configuration such as memory configuration, logging options, proxy server configuration or caching configuration.
  - A wide range of environment or business-specific configuration such as connecting to SharePoint for the MashZone NextGen Add-On for SharePoint or managing provide and category taxonomies.
- Configuration for authentication and authorization. See "MashZone NextGen Security" on page 1602 for instructions.
- Other MashZone NextGen and MashZone Repositories tasks.
- Administration tasks are discussed in MashZone NextGen Server Administration for:
  - Viewing and managing server logs.
  - Managing resource files.
  - Migrating to new MashZone NextGen releases.
  - Deploying MashZone NextGen instances and artifacts.
  - Clustering MashZone NextGen Servers.

MashZone NextGen Security

MashZone NextGen provides control of user interactions to register or create mashable information sources, mashups and apps and secure access for all users to work with these artifacts based on policies that you define. Security is embodied in these layers:



- Change password: For reasons of security we strongly recommend that the MashZone NextGen administrator should change the standard MashZone NextGen password after installation. See "Change technical user password" on page 1603.
- **User Authentication**: based on the protocols shown above. You can also allow anonymous access if needed. See "Authentication and Guest Access" on page 1604 for details.
- **Authorization Policies**: to determine the actions that users can perform with mashables, mashups and apps. Policies also determine user access to the features and tools in MashZone NextGen Hub and the MashZone NextGen Enterprise AppDepot. See "Authorization Policies and Permissions" on page 1619 for details.

- **Security Profiles**: that define the requirements for secure communication with mashable information sources.

  MashZone NextGen supports the well-known protocols shown above. MashZone NextGen developers can also create custom security profiles to support mashable information sources with unique requirements. See "Configure Secure Connections for Mashables" on page 164 for more information.

- **Feature Security**: to control any features in the MashZone NextGen platform that have security implications, such as scripting access in mashups, or that may conflict with the security requirements of your organization. See "Disable Mashup Features" on page 1691 and "Configure the Default Operations Generated for Database Mashable" on page 1684 for more information.

Please consider the following security-relevant aspects :

- Always keep your operating system, installed components and applications updated. Run necessary security updates on a regular basis, in particular for your Web-Browser and installed plug-ins.
- Always keep your MashZone NextGen installation updated. Regularly check if new fixes are available for your installation and install them.
- To prevent unauthorized access to your system, only a limited number of users should be granted direct system access (e.g., remote RDP access or directly via a management console).
- Limit network access by operating the server components behind a firewall. Only necessary services should be open in the firewall (e. g. database).
- Hide network ports used solely for internal communication between server components.

Change technical user password

For reasons of security we strongly recommend that the MashZone NextGen administrator should change the standard technical user password after installation. The technical user password is encrypt and stored in two modules. You have to change both occurrences.

<*MashZoneNG-install*> /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/mz.properties

<*MashZoneNG-install*> /apache-tomee-jaxrs/webapps/mashzone/WEB-INF/mashzone.properties

Note: This procedure is only required for MashZone NextGen 3.9.01.

Procedure

1. Change the password in .../mz.properties.

a. Encrypt a new password using the `padmin` tool. Open the command line and enter following command. Replace the variable <password> by your new password, e.g. newPassword.

```
$ <MashZoneNG-install>/prestocli/bin/padmin encryptProperty -u
Administrator -w manage -p <password>
```

b. Copy the output of the command line into mz.properties, e.g. {ENC}A +yyI2FYYBy33lgNCWGQIQ==.

mzServer.secrete={ENC}A+yyI2FYYBy33lgNCWGQIQ==

2. Change the password in .../mashzone.properties.

a. Encrypt a new password using the `encryptpassword` tool. Open the command line and enter following command. Replace the variable <password> by your new password, e.g. newPassword.

```
$ <MashZoneNG-install>mashzone/tools/runtool encryptpassword -
password <password>
```

b. Copy the output of the command line into mashzone.properties, e.g. 46f712a61dc8d7ed244bf0ffd266ae1e.

presto.basicAuthPassword=46f712a61dc8d7ed244bf0ffd266ae1e

The technical user password is changed.

Authentication and Guest Access

MashZone NextGen accepts requests from *both* unauthenticated (guests) and authenticated users. Guests, however, can only access mashables, mashup and apps in MashZone NextGen that explicitly allow guest access.

Authentication is required:

- To access the MashZone NextGen Hub, the AppDepot or the MashZone NextGen Mobile apps for mobile phones or mobile tablets. Access to specific mashables, mashups, apps and specific features in MashZone NextGen Hub or the AppDepot is determined by the user's permissions.
- To use any feature in any MashZone NextGen Add-On that accesses the MashZone NextGen Server, unless that Add-On also supports guest access.

  For example, portal users may view MashZone NextGen apps or mashups using the MashZone NextGen Add-On for Portals if those artifacts and the portal supports guest access. If guest access is not enabled, authentication is required to see information on MashZone NextGen apps or mashups.
- To use apps or other web applications outside of MashZone NextGen Hub, the AppDepot or the MashZone NextGen Mobile apps, unless all of the apps, mashables or mashups that are used explicitly permit guest access. Artifacts that permit guest access have *no* authorization requirements.

Requests are rejected with an authentication error when they do not provide one of:

- A valid MashZone NextGen session cookie. Sessions that have timed out are rejected with an appropriate error. See "Sessions and Timeouts" on page 1606 for more information.
- Valid credentials. See "Valid Credentials" on page 1605 for more information.
- Guest access header or parameter information. See "Enabling Guest Access" on page 1606 for more information.

User Authentication

MashZone NextGen is initially installed with a set of "Default User Accounts" on page 1607 that you can use to get started. You configure MashZone NextGen to work with your LDAP Directory or you can continue to use the Default User Repository and simply add users and user groups to MashZone NextGen. See "Use the Default MashZone NextGen User Repository" on page 1597, "Manage Users" on page 1597 and "Manage User Groups" on page 1599 for more information.

Authentication to verify user identities is performed against LDAP or the default User Repository and uses one of these protocols:

- Basic authentication with username and password

  This is the default authentication mechanism. No additional configuration is needed.

- SSL and User Certificates

  See "Authentication with Digital Certificates/SSL" on page 1614 for configuration instructions.

- A configurable Single Sign-On solution

  See "Authentication with Single Sign-On Solutions" on page 1607 for configuration instructions.

Permission to work with apps and other MashZone NextGen artifacts can also be granted to guests (unauthenticated users), if needed.

Valid Credentials

When authentication is required, requests must have a valid MashZone NextGen session for an existing authenticated user or must supply either user credentials or digital certificate for authentication or an SSO token or ticket for a user that has been authenticated by the SSO solution. MashZone NextGen uses certificates, tokens or tickets to obtain the user's identity.

MashZone NextGen supports the following mechanisms to obtain user credentials or user IDs:

- Basic authentication using username and passwords. This is authenticated against the MashZone NextGen User Repository which may be a database or your LDAP

Directory. See "Use the Default MashZone NextGen User Repository" on page 1597 for more information.

> **Note:** This is the *only* mechanism for obtaining user credentials that is supported by the MashZone NextGen Mobile apps.

- SSL and Certificate authentication where the user identifier in certificate information is configurable. This is authenticated against the MashZone NextGen User Repository which may be a database or your LDAP Directory, *unless* Dynamic User Support is enabled. See "Use the Default MashZone NextGen User Repository" on page 1597 and "Authentication with Digital Certificates/SSL" on page 1614 for more information.
- Single Sign-On (SSO) solutions which are configurable. With SSO enabled, MashZone NextGen delegates authentication to the SSO solution. Typically, configuration identifies an SSO token, ticket or cookie that MashZone NextGen uses to verify authentication with the SSO solution and to obtain the user ID. See "Authentication with Single Sign-On Solutions" on page 1607 for more information.

If an authenticated request has no MashZone NextGen session, MashZone NextGen starts a new session and generates a MashZone NextGen session cookie. See "Sessions and Timeouts" on page 1606 for more information.

Sessions and Timeouts

MashZone NextGen is based on the standard J2EE session mechanism supported by your application server. MashZone NextGen maintains a separate HTTP session for each authenticated user that has a unique session cookie. Each request with a valid MashZone NextGen session cookie extends the timeout limit for that user session.

SSO solutions maintain their own sessions and may use their own session cookies. SSO session cookies can be used to authenticate users in MashZone NextGen. SSO sessions and MashZone NextGen session are separate.

The default session timeout for MashZone NextGen is 30 minutes, defined in the *web-apps-home* /presto/web.xml configuration file. In general, HTTP session timeouts can be configured in web.xml, unless the application server provides other configuration mechanisms. Please see your application server documentation for additional information on session configuration.

Enabling Guest Access

To allow unauthenticated users to work with apps or other artifacts outside of MashZone NextGen, you must enable guest access to both:

- That app
- Every mashable, mashup or app that the app or other artifact uses

Simply add the built-in `MashZone_NextGen_Guest` user group in the run permissions for those artifacts. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.

Default User Accounts

MashZone NextGen has four default user accounts that you can use 'out-of-the-box' to access MashZone NextGen Hub, the AppDepot and the MashZone NextGen Modile apps. These default users also illustrate the basic permissions to features in MashZone NextGen. See "Built-In MashZone NextGen User Groups and Permissions" on page 1622 for more information on permissions.

Username	Password	Built-in Group / Permissions	Description
`Administrator`	`manage`	`MashZone NextGen_Administrato`	A MashZone NextGen administrator.
`dev`	`devdev`	`MashZone NextGen_Developer`	A developer.
`power`	`powerpower`	`MashZone NextGen_PowerUser`	A domain expert or power user.
`user`	`useruser`	`MashZone NextGen_AuthenticatedUser`	An end user or any user in the MashZone NextGen User Repository.

If you configure MashZone NextGen to use your LDAP Directory as the MashZone NextGen User Repository, these default user accounts are automatically disabled. If you use the Default User Repository, you can delete these user accounts in the Admin Console.

Important: You must make sure that at least one user account has MashZone NextGen administrator permissions.

Authentication with Single Sign-On Solutions

With a single sign-on (SSO) solution, MashZone NextGen delegates authentication to the SSO layer. MashZone NextGen has the following pre-configured options to integrate with SSO solutions:

- SharePoint and the MashZone NextGen Add-On for SharePoint.

  Important: Single sign-on authentication with SharePoint and MashZone NextGen Add-On for SharePoint is enabled by default in MashZone NextGen. No additional configuration is needed in MashZone NextGen to integrate this token-based SSO solution.

- Agent-based SSO solutions, such as Netegrity SiteMinder. See "Configuration for Agent-Based SSO Solutions" on page 1608 for instructions.

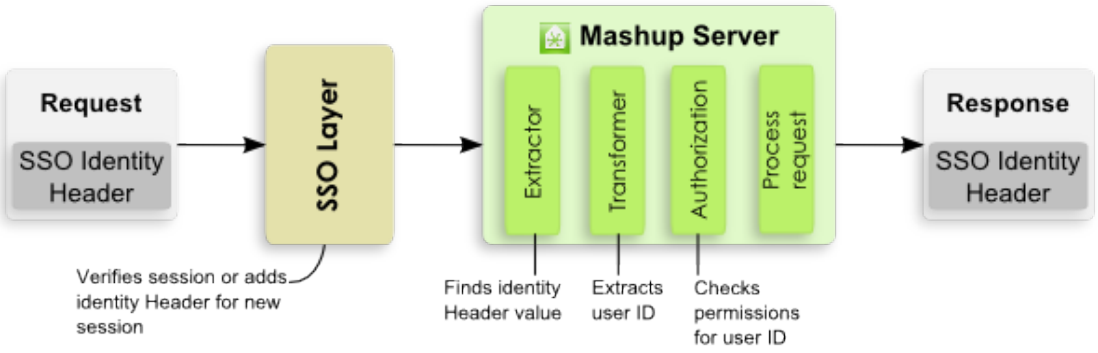
  Note: The MashZone NextGen Add-On for SharePoint is not fully compatible with agent-based SSO solutions. Functionality in MashZone NextGen Hub is available with an agent-based SSO solution. However, known issues limit the functionality available in SharePoint in this case. See "Authentication Designs for P4S 2010" on page 1373 for more information.

- The Central Authentication Service (CAS) using the CAS2 protocol. CAS is a ticket-based SSO solution. It also supports authentication proxies with CAS2, which enables MashZone NextGen to use CAS for SSO and also work with the CAS Security Profile for mashable information sources.

  See "Configuration for the CAS SSO Solution" on page 1611 for instructions.

Configuration for Agent-Based SSO Solutions

With agent-based SSO, the basic flow of authentication and user identity information looks something like this:



MashZone NextGen delegates authentication to the SSO layer, but expects user identity information from the SSO layer in the request in either an HTTP header or a parameter in the request URL. MashZone NextGen uses an extractor to find identity information in the header or parameter, and uses a transformer, to derive the user ID from the identity information. MashZone NextGen then uses the user ID to perform authorization and process the request.

To configure MashZone NextGen to work with an agent-based SSO layer, you configure the extractor and the transformer layers to work with your SSO solution and the identity information for your environment. MashZone NextGen provides a default extractor that looks for an HTTP header or parameter by name. MashZone NextGen also provides default transformers that handles cases where the identity information is just the user ID or can be found within the identity information using a regular expression.

Note: You can also implement custom extraction or transformation layers to integrate MashZone NextGen with your SSO solution. See "Implementing a Custom SSO Filter" on page 1614 for details.

1. If needed, configure the MashZone NextGen User Repository. See "Use the Default MashZone NextGen User Repository" on page 1597 for more information.

   In previous releases, MashZone NextGen only supported SSO solutions with LDAP as the MashZone NextGen User Repository. This restriction no longer applies.

2. Change the SSO filter in the applicationContext-security.xml configuration file for the MashZone NextGen Server:

   a. Open applicationContext-security.xml in any text or XML editor.

      This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.

   b. Comment out the SSO filter bean (`<bean id="ssoProcessingFilter">`) for SharePoint (`class="com.jackbe.jbp.sas.security.ui.sso.sp.SharepointSSOFilter">`).

      For example:

```
<!-- <bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    ...
</bean> -->
```

   c. Uncomment the SSO filter bean (`<bean id="ssoProcessingFilter">`) for agent-based solutions (`class="com.jackbe.jbp.sas.security.ui.sso.SSOPreAuthenticatedFilter"`).

      For example:

```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    ...
</bean>
```

3. In the agent-based SSO filter bean, configure the `principalExtractor` property:

   - The default extractor uses a bean with the `HttpHeaderOrParamTokenExtractor` class.

```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    <property name="principalExtractor">
      <bean
>
        <property name="httpHeaderName" value="SM_USER"/>
      </bean>
```

```
    </property>
    ...
</bean>
```

Change the value of the `httpHeaderName` property for this extractor bean to the name of the HTTP header or parameter that contains user identify information from your SSO solution.

- If you have a custom extractor class, replace the default extractor bean with configuration for your custom class.

4. In the agent-based SSO filter bean, configure the `principalTransformer` property:

   - The default transformer property uses a bean with the `RegexExtractionStringTransformation` class. This uses a regular expression to extract some portion of the value for the SSO header or parameter to get the final user ID that MashZone NextGen can use for authorization checks.

```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    <property name="principalExtractor">
      <bean
>
        <property name="httpHeaderName" value="SM_USER"/>
      </bean>
    </property>
    <property name="principalTransformation">
      <bean
>
        <constructor-arg index="0" value="CN=(.*?),"/>
      </bean>
    </property>
</bean>
```

If the value of the SSO solution header or parameter contains more than just the user ID, for example a full DN from LDAP for a user, you can change the regular expression in the <constructor-arg/> parameter for the default bean to extract the user ID. The default regular expression extracts the CN portion of a user DN from an LDAP Directory.

If the value of the SSO solution header or parameter is *just* the user ID, no further transformation is needed. Change the `principalTransformer` bean to do nothing using the `NoOpStringTransformation` bean:

```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    <property name="principalExtractor">
      <bean
>
        <property name="httpHeaderName" value="SM_USER"/>
      </bean>
    </property>
    <property name="principalTransformation">
      <bean
/>
```

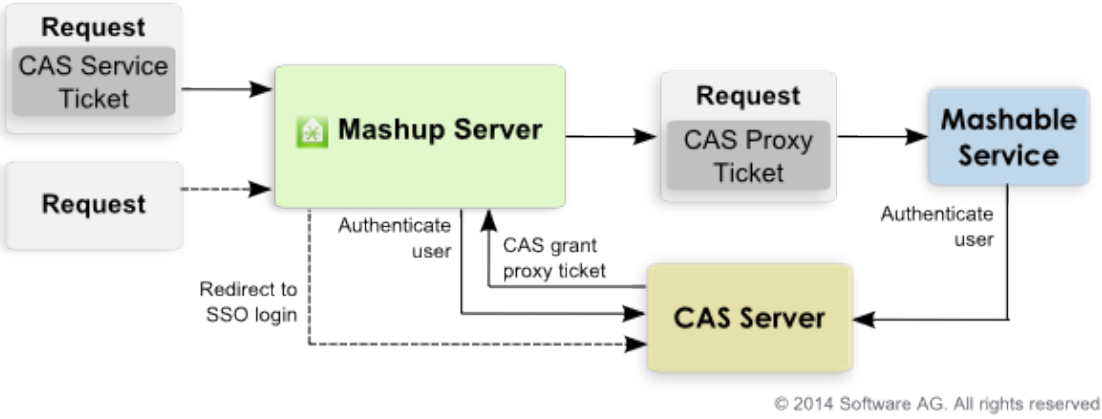
```
    </property>
</bean>
```

- If you have a custom transformation class, replace the default transformer bean with configuration for your custom class.

5. Save this file and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Configuration for the CAS SSO Solution

CAS uses tickets in requests that 'secured services' can use to validate the user is authenticated, as shown below:



When users access MashZone NextGen, if they have logged in with CAS, the request includes a service ticket unique to that user. MashZone NextGen validates this ticket to retrieve user ID information needed for authorization.

MashZone NextGen also obtains a proxy granting ticket from CAS to use if the user runs a mashable that is also secured by CAS. This proxy feature allows MashZone NextGen to send the mashable a proxy ticket that the mashable can use to authenticate the user.

If users access MashZone NextGen without first logging in with CAS, MashZone NextGen redirects users to the login page for CAS instead of the default MashZone NextGen login page.

To configure MashZone NextGen authentication using CAS, handle login redirects and enable CAS security profiles for mashables:

1. Enable HTTPS for communication between the MashZone NextGen Server and the CAS Server. You must:

   a. Configure the application server hosting the MashZone NextGen Server to listen to separate ports for HTTP and for HTTPS. In addition, you must configure a certificate store for the application server.

      See "Configure HTTPS and Certficate Stores in the Application Server" on page 1648 for instructions for Tomcat. If MashZone NextGen is deployed in another

application server, see documentation for your application server for more information.

b. Obtain a certificate for the MashZone NextGen Server and add it to the certificate store.

   If the CAS Server uses a self-signed certificate, you must also add this to the certificate store.

   See "The Certificate Store and Certificates" on page 1645 for more information.

2. Open applicationContext-security.xml in any text or XML editor.

   This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.

3. Make sure that the import statement for applicationContext-security-authn-cas2.xml is uncommented.

   For example:

```
...
 <import resource="applicationContext-security-authn-rememberme.xml"/>
 <import resource="applicationContext-scheduler.xml"/>
 <!-- import resource="applicationContext-security-authn-x509.xml"/-->
 <!-- import resource="applicationContext-security-authn-rsa.xml"/-->
<import resource="applicationContext-security-authn-cas2.xml"/>
...
```

4. Find the bean with `authenticationEntryPointFilter` ID and change the value of the `defaultAuthenticationModuleName` property to `cas`.

   For example:

```
...
  <bean id="authenticationEntryPointFilter"
>
    <property name="authentiationModules">
      <map>
        <entry key="cas" value-ref="casAuthenticationEntryPoint"/>
        <entry key="prestohub"
          value-ref="prestoDefaultAuthenticationEntryPoint"/>
      </map>
    </property>
<property name="defaultAuthenticationModuleName" value="cas"/>
  </bean>
...
```

5. Find the bean with `preauthAuthProvider` ID and:

   a. Comment out the preauthenticated user details property based on `UserDetailsByNameServiceWrapper`.

   b. Uncomment the preauthenticated user details property based on `casAuthenticatedUserDetailsService`.

   For example:

```
  <bean id="preauthAuthProvider"
class="org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvider">
<property name="preAuthenticatedUserDetailsService"
ref="casAuthenticationUserDetailsService"/>
    <!-- property name="preAuthenticatedUserDetailsService">
```

```
      <bean id="userDetailsServiceWrapper"
class="org.springframework.security.userdetails.UserDetailsByNameServiceWrapper">
        <property name="userDetailsService" ref="userRepositoryAccessAdapter"/>
      </bean>
    </property -->
  </bean>
```

6. Save your changes to applicationContext-security.xml.
7. Open applicationContext-security-filters-default.xml in any text or XML editor.

   This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.
8. Make sure that the line beginning with `/**/cas/**` is not commented out. Save your changes, if any.
9. Set configuration properties to redirect users to the CAS login form if requests attempt to access MashZone NextGen directly without a valid CAS ticket. You must:
   a. Open the sso.properties file in any text editor.

      This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.
   b. Set the following properties for the MashZone NextGen Server:
      - `prestoServerInfo.host` = the host name or IP address for this MashZone NextGen Server.
      - `prestoServerInfo.httpPort` = the HTTP port for this MashZone NextGen Server. This is 8080 if you installed MashZone NextGen with default ports.
      - `prestoServerInfo.httpsPort` = the HTTPS port for this MashZone NextGen Server. For Tomcat, 8443 is the default HTTPS port.
   c. Set the following properties for the CAS Server:
      - `ssoServerInfo.host` = the host name or IP address for this CAS Server.

        If the CAS server is deployed at `https://cas.myOrg.com:9443/cas`, for example, the host would be `cas.myOrg.com`.
      - `ssoServerInfo.httpsPort` = the HTTPS port for this CAS Server.

        If the CAS server is deployed at `https://cas.myOrg.com:9443/cas`, for example, the HTTPS port would be `9443`.
      - `ssoServerInfo.rootPath` = the relative path, starting from the host and HTTPS port for this CAS Server.

        If the CAS server is deployed at `https://cas.myOrg.com:9443/cas`, for example, the root path would be `cas`.
      - `ssoServerInfo.loginPath` = the relative path, starting from the root path where this CAS server is deployed, to the login page where users should be redirected if they do not have a valid CAS ticket.

        If the URL for your CAS login page is `https://cas.myOrg.com:9443/cas/login`, this property should be `login` as the rest of the URL is set in other properties.

d. Save your changes.

10. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Implementing a Custom SSO Filter

If the default extractor and transformer filters available in MashZone NextGen do not provide the functionality needed to allow MashZone NextGen to work with your SSO solution, you can create custom filters using the MashZone NextGen SSO Filter API.

To use this API

1. Add the following JARs and classes to your classpath:
   - Classes in the *web-apps-home* /presto/WEB-INF/classes folder.
   - The *web-apps-home* /presto/WEB-INF/lib/presto_common.jar file.
2. Implement one or both filters:
   - To create a custom extractor, implement the `SSOTokenExtractor` interface, typically using the `AbstractSSOTokenExtractor` base class.
   - To create a custom transformer, implement the `Transformation` interface.
3. Add these classes to the classpath. Copy either the compiled class file or a JAR containing the compiled class file to one of these folders, respectively:
   - The external configuration folder, if any, for the MashZone NextGen Server. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

     Important: Deploying additional resources, such as custom SSO filters, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

   - *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
   - *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

Authentication with Digital Certificates/SSL

There are two aspects of authentication for MashZone NextGen that you can configure for digital certificates: 1) whether MashZone NextGen accepts certificates for user authentication and 2) what information MashZone NextGen uses from the certificates to perform authentication.

Certificate authentication in MashZone NextGen uses Personal Digital Certificates (PDC) from a client. The default authentication process when MashZone NextGen receives a

certificate looks for a user ID in the CN portion of the certificate's subjectDN. This user ID is authenticated against the User Repository.

If it is a valid user ID, this ends authentication. MashZone NextGen continues with authorization for the request. If the user ID is not valid, the request is rejected.

To enable authentication based on digital certificates

1. Configure the MashZone NextGen Server to use mutual SSL. See "Configure MashZone NextGen for SSL and Digital Certificates" on page 1643 for instructions.
2. Using any text or XML editor, edit the applicationContext-security.xml file in the *web-apps-home* /presto/WEB-INF/classes directory and:
   a. Remove the comment markers from the `<import>` statement for the applicationContext-security-authn-x509.xml file.

      The configuration would look something like this:

      ```
      <beans>
        <import resource="applicationContext-security-authn-rememberme.xml" />
        <import resource="applicationContext-security-scheduler.xml" />
        <import resource="applicationContext-security-authn-x509.xml" />
        <!--<import resource="applicationContext-security-authn-rsa.xml" /> -->
      ...
      </beans>
      ```

   b. Save your changes to this file.
3. If needed, change the default certificate authentication behavior with one or more of these options:
   - Configure Alternate User ID Extraction to change where MashZone NextGen obtains the user ID.
   - Configure Dynamic User Support to enable MashZone NextGen to accept certificates for user IDs not found in the User Repository.
   - Configure Additional Certificate Validation beyond simple user IDs.
4. Enable certificate authentication for the MashZone NextGen REST API. See "Configure the MashZone NextGen REST API to Use Certificate Authentication" on page 1616 for instructions.
5. If needed, enable certificate caching for the MashZone NextGen Server.

   By default, the MashZone NextGen Server does not cache user certificates. This ensures that any changes to user identification or authorization are detected as soon as possible but can impact performance. To turn caching on:
   a. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
   b. Find the `x509AutheticationProvider` bean.
   c. Add `<property name="certificateCachingEnabled" value="true" />` to the list of properties for this bean.

d. Save your changes to this file.

6. To apply these changes, restart the MashZone NextGen Server.

Configure the MashZone NextGen REST API to Use Certificate Authentication

By default, certificate authentication is *not* enabled for the REST API to MashZone NextGen or for MashZone NextGen Connect for JavaScript (PC4JS).

1. Using any text or XML editor, edit the applicationContext-security-filters-default.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
2. Find the Filter Chain Proxy (`<bean id="filterChainProxy">`) and:
   a. Find the line for `/**/api/rest/**`.
   b. Add `x509ProcessingFilter,` *after* `restLoginProcessingFilter`
   c. In this same bean, find the line for `/**/api*`.
   d. Add `x509ProcessingFilter,` *after* `jumpLoginProcessingFilter`

   The result should look something like this:

```
<bean id="filterChainProxy"
>
  <property name="filterInvocationDefinitionSource">
    <value>
      CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
      PATTERN_TYPE_APACHE_ANT
      /**/esd/api/mashsoap/**=basicReqFlowSupportFilter, sessionContextIntegrationFilter, shar
      soapRequestAuthenticationFilter, basicProcessingFilter,  anonymousProcessingFilter, exce
      /**/edge/api/mashsoap/**=basicReqFlowSupportFilter,  sessionContextIntegrationFilter, sh
      soapRequestAuthenticationFilter, basicProcessingFilter,  anonymousProcessingFilter, exce
      /**/api/soap/**=basicReqFlowSupportFilter,  sessionContextIntegrationFilter, sharepointS
      wsSecurityProcessingFilter, basicProcessingFilter, anonymousProcessingFilter, exceptionT
     /**/api/rest/**=restReqFlowSupportFilter,  sessionContextIntegrationFilter, sharepointSSO
      restLoginProcessingFilter, x509ProcessingFiler, basicProcessingFilter,  anonymousProcess
      sessionTimeoutDetectionFilter, exceptionTranslationFilter
     /**/emml/debug=basicReqFlowSupportFilter, sessionContextIntegrationFilter,ssoProcessingFi
     restLoginProcessingFilter, basicProcessingFilter, anonymousProcessingFilter, exceptionTra
     /**/api*= jumpReqFlowSupportFilter, sessionContextIntegrationFilter,ssoProcessingFilter,j
     jumpLoginProcessingFilter, x509ProcessingFiler, basicProcessingFilter, anonymousProcessin
     sessionTimeoutDetectionFilter, exceptionTranslationFilter,filterInvocationInterceptor,ses
      ...
```

   This configuration allows both the default HTTP connections with user credentials and HTTPS with digital certificates.

3. Save this change.
4. Open the applicationContext-security.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
5. Find the REST Login Processing Filter (`<bean id="restLoginProcessingFilter">`) and add `<property name="ignoreFailure" value="true" />`.

   The bean configuration should look like:

```
<bean id="restLoginProcessingFilter"
>
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="sessionManager" ref="sessionManager" />
  <property name="rememberMeServices" ref="rememberMeServices"/>
  <property name="ignoreFailure" value="true"/>
</bean>
```

6. Find the Authentication Manager (`<bean id="authenticationManager">`) and uncomment the reference to the `x509AuthenticationProvider`.

   The bean configuration should look like:

```
<bean id="authenticationManager"
>
  <property name="providers">
    <list>
      <ref bean="x509AuthenticationProvider"/>
      <ref local="preauthAuthProvider"/>
      <ref local="adminAuthenticationProvider"/>
      <ref bean="defaultAuthenticationProvider"/>
      <ref bean="rememberMeAuthenticationProvider"/>
      <ref local="anonymousAuthenticationProvider"/>
    </list>
  </property>
</bean>
```

7. Save your changes to this file.

Configure Alternate User ID Extraction

You can use regular expressions to define other portions of the certificate's subjectDN as the source for the user's ID. To define an alternate location for the user ID:

1. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
2. Find the x509 Authorities Populator (`<bean id="x509AuthoritiesPopulator" >`) and:
   a. Remove the comment markers around the `<property name="subjectDNregex">` element.
   b. Change the regular expression in the `<value>` to define what to use as the user ID.
3. Save your changes to this file.

Configure Dynamic User Support

Dynamic user support allows MashZone NextGen to accept a valid certificate even if the user is not provisioned in the User Repository. You can also define a set of roles for dynamic users to enable specific service access. To configure dynamic user support:

1. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.

2. Find the x509 Authorities Populator (`<bean id="x509AuthoritiesPopulator" >`) and:

   a. Change the value attribute for `<property name="dynamicUser"/>` to `IN_MEMORY`.

      For example:

      ```
      <property name="dynamicUser" value="IN_MEMORY"/>
      ```

   b. To set up groups or roles to use as authorization for dynamic users, add or uncomment `<property name="dynamicUserRoles">`.

   c. Add a `<list>` child and add a `<value>` child under that for each group or role that dynamic users should be authorized for.

      For example:

      ```
      <property name="dynamicUserRoles">
        <list>
          <value>EditPreferences</value>
          <value>ViewNews</value>
        </list>
      </property>
      ```

3. Save your changes to this file.

Configure Additional Certificate Validation

You can have certificate authentication perform additional validation beyond simple user ID checks.

1. Implement the additional validation logic in a class that implements the com.jackbe.jbp.sas.security.x509.x509CertValidator interface.

   To do this, add the following JARs and classes to your classpath:

   - Classes in the *web-apps-home* /presto/WEB-INF/classes folder.
   - The *web-apps-home* /presto/WEB-INF/lib/presto_common.jar file.

   See the *Custom Certificate Validation API* for details on implementing this interface.

   Then add your custom class to the classpath in one of these folder:

   - The external configuration folder, if any, for the MashZone NextGen Server. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

     Important: Deploying additional resources, such as custom validation classes, to an external configuration folder simplifies future deployments or MashZone NextGen Server clusters.

   - *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

- *web-apps-home* /presto/WEB-INF/lib. TThis is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

2. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
3. Find the x509 Authentication Provider (`<bean id="x509AuthenticationProvider" >`) and:
   a. Find the <property name="validators"> element.
   b. Add a <list> child and add a <bean> child with your implementation class name.

      For example:

```
<bean id="x509AuthenticationProvider">
    ...
    <property name="validators">
      <list>
        <bean/>
      </list>
    </property>
    ....
</bean>
```

4. Save your changes to this file.

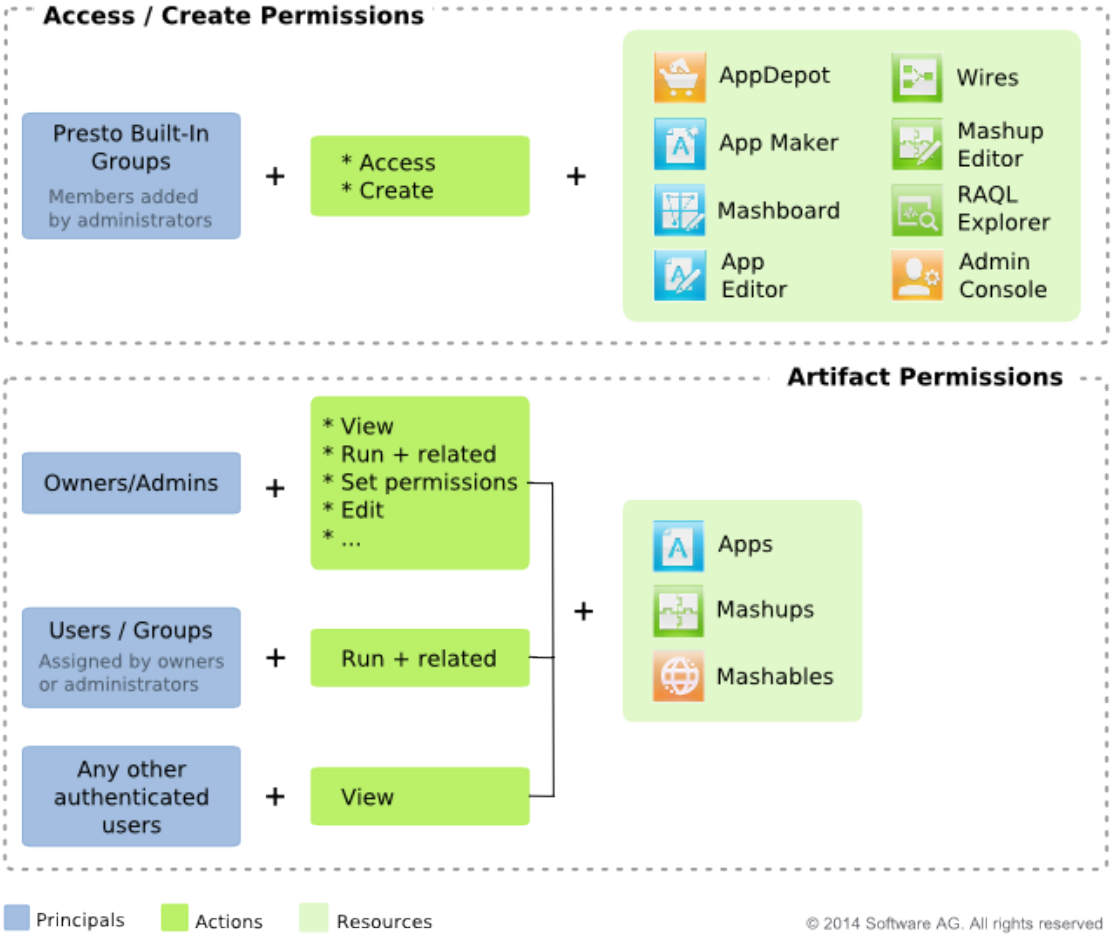
Authorization Policies and Permissions

Authorization policies determine the actions that users can perform with the mashables, mashups and apps that MashZone NextGen governs. Policies also determine user access to the features and tools in the MashZone NextGen Hub and the MashZone NextGen Enterprise AppDepot.

By default, authorization is enabled in MashZone NextGen. All actions are forbidden unless explicitly granted in a policy.

Note: You can choose to disable authorization during an initial development phase to simplify access to register and create mashables, mashups and apps. See "Enable or Disable Authorization" on page 1625 for instructions.

The categories of authorization policies that are defined in MashZone NextGen are shown below.



- **Access/Create Permissions**: are defined using MashZone NextGen built-in user groups as the principals. See the "Built-In MashZone NextGen User Groups and Permissions" on page 1622 topic for detailed information these policies.

  To grant access to MashZone NextGen tools and enable users to create artifacts in MashZone NextGen Hub, you add users to these built-in groups. See "Grant User Access to MashZone NextGen with Built-in Groups" on page 1621 for instructions.

- **Owner/Admin Permissions**: users automatically obtain owner permissions when they create artifacts. Administrator permissions are defined when you assign users to the `MashZone NextGen_Administrator` built-in group (see Access/Create policies).

  Owners have full permissions to all actions for the artifacts they create, *except* the feature/unfeature action. Administrators have owner permissions for *all* artifacts as well as for the feature/unfeature action.

- **Run Permissions**: owners and MashZone NextGen administrators grant run permissions to other users to allow them to use that artifact. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions."Automatically Grant Run Permissions to Users and Groups" on page 1624. For mashups and

apps, users must also have run permissions for the other mashable information sources, mashups or apps that are used by that mashup or app.

You can also grant guest access to use artifacts. Guest access grants permission for anyone to run that artifact, even users who are not logged in. See "Authentication and Guest Access" on page 1604 for instructions.

Users also get several other related permissions when you grant run permissions. See the "Built-In MashZone NextGen User Groups and Permissions" on page 1622 topic for more information on the additional permissions granted with run.

- **View Permissions**: authenticated users can see artifacts in MashZone NextGen Hub and the AppDepot even for artifacts for which they do not have run permissions. They can open the artifact and request permissions, but they cannot run or preview the artifact.

  You can also restrict view permissions. See "Set View Permissions with a Search Filter" on page 1625 for information.

Grant User Access to MashZone NextGen with Built-in Groups

All users in the MashZone NextGen User Repository automatically belong to the `MashZone NextGen_AuthenticatedUsers` built-in group which has permission to access the MashZone NextGenAppDepot and work with any apps to which they also have been granted run permissions. To enable users to work in MashZone NextGen Hub to find, register or create mashables, mashups and apps, you must add them to the `MashZone NextGen_PowerUser`, `MashZone NextGen_Developer` or `MashZone NextGen_Administrator` groups.

See the "Built-In MashZone NextGen User Groups and Permissions" on page 1622 topic for information on the specific access policies for these groups. Or use the "Default User Accounts" on page 1607 in MashZone NextGen to better understand the permissions for these groups.

- If you are using the Default User Repository with MashZone NextGen, both groups and users are defined with the Admin Console. To grant users permissions with the MashZone NextGen built-in groups:

  1. Add users to the MashZone NextGen Repository. See "Create Users" on page 1598 for instructions.
  2. Assign users to the appropriate built-in groups. See "Edit, Grant Permissions and other User Management Tasks" on page 1598 for instructions.
  3. If desired, you can also automatically add users as members to groups when you create users. See "Automatically Assign New Users to Groups" on page 1600 for instructions.

- If you have configured MashZone NextGen to use your LDAP Directory as the User Repository, you relate users to the MashZone NextGen built-in groups in LDAP. To grant users permissions with the MashZone NextGen built-in groups:

1. Add `MashZone NextGen_Administrator`, `MashZone NextGen_Developer` and `MashZone NextGen_PowerUser` as new groups in LDAP.

   Note: To map users and groups in LDAP to MashZone NextGen built-in permissions, you add these predefined names to your LDAP Directory. Mapping from configuration in MashZone NextGen based on LDAP attributes is possible. Or defining alias names for these built-in groups is also possible. For more information and assistance, please contact your Software AG sales representative.

2. Assign users to these new groups in LDAP.

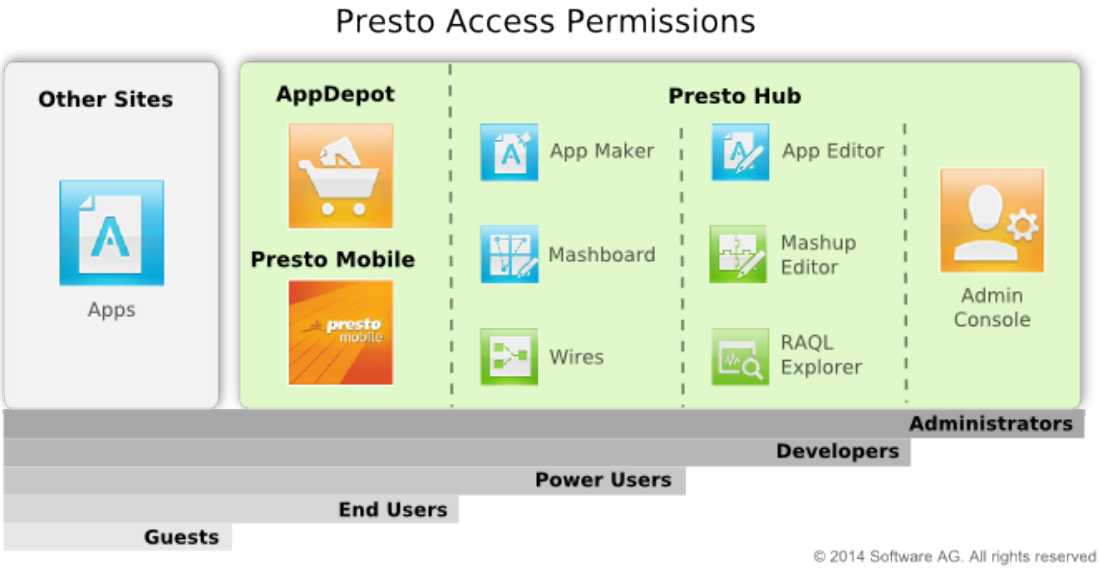
Built-In MashZone NextGen User Groups and Permissions

MashZone NextGen has a set of built-in user groups that define access permissions to the various features in MashZone NextGen Hub and the AppDepot. These built-in groups also define permissions for all artifact actions *except* for permissions to run mashables, mashups or apps.

For more details on the permissions for these built-in groups, see "Access Policies Using MashZone NextGen Built-In Groups" on page 1622 and "Artifact Permissions for Users with Run Permissions" on page 1623.

Access Policies Using MashZone NextGen Built-In Groups

A high-level view of access policies for the built-in groups is shown below:



- *Guests* = users in other sites who are not authenticated. Guests can work with apps deployed in other sites if the app and all other artifacts that it depends on have

granted run permissions to the `MashZone NextGen_Guest` built-in group. See "Enabling Guest Access" on page 1606 for instructions.

The most common use is to allow apps to run in public web sites or other environments where secure access is not needed.

> **Note:** Granting guest access to mashables, mashups and apps also implicitly grants run permissions to the artifact to any authenticated MashZone NextGen user in the AppDepot and in MashZone NextGen Hub.

- *End Users* = all authenticated users (in the MashZone NextGen Repository) that are not in another built-in group. Authenticated users can access MashZone NextGen Hub and the AppDepot to find artifacts, but they can only use the artifacts to which they have been granted run permissions. They also have *no access* to tools that create artifacts.
- *Power Users* = users in the `MashZone NextGen_PowerUser` group can register mashables and create mashups or apps using wizards or other visual tools in MashZone NextGen Hub. Power users cannot use tools or other features that are highly technical or that require coding with EMML, RAQL, the App Specification or other MashZone NextGen APIs or extension points.

  This group is typically used for domain experts, business analysts or other non-technical users who should be able to create artifacts using wizards or visual tools.
- **Developers** = users in the `MashZone NextGen_Developer` group can find, register and create mashables, mashups or apps using both visual tools and code editors that use the full power of EMML, RAQL, the App Specification and other MashZone NextGen extension points. Developers also have access to other technical information, such as the Technical Specification for mashables and mashups or the API Console.

  This group is typically used for IT or line-of-business developers involved in developing apps, mashups or mashables for specific projects. Developers may also develop other extension features to provide specific capabilities in MashZone NextGen Hub for power users.
- **Administrators** = users in the `MashZone NextGen_Administrator` group have unrestricted permissions in MashZone NextGen. They can work with any tools, features or artifacts. They also have permissions to use the Admin Console to configure and manage MashZone NextGen and to approve apps that have been submitted to the AppDepot.

Administrators are the only built-in group that is required. You can use the other built-in groups to grant access to specific MashZone NextGen tools and features.

Artifact Permissions for Users with Run Permissions

With artifacts, owners and administrators have full permissions for all actions, subject to filtering for membership in power user or developer groups. Authenticated users can only see artifacts in search results or activities. The fourth and final group with artifact permissions are users who have been granted run permissions.

Granting run permissions allows users to run a mashable or mashup to see results or to preview an app. Run permissions also automatically grants permissions for other actions including:

- Comment, rate and tag the artifact.
- Add, delete and manage views for mashables or mashups.
- Take, view and schedule snapshots for mashables or mashups.
- Create basic apps or mashups from mashables or mashups.
- Publish apps to SharePoint or embed apps in other environments. Only *owners*, however, can publish an app to the AppDepot.
- View the technical specification for the artifact or view dependencies (related items).

Automatically Grant Run Permissions to Users and Groups

MashZone NextGen administrators can define one or more groups or users that are automatically granted run permissions when users register mashables or create mashups or apps.

These default run permissions are automatically added to all new artifacts of that type, but can be deleted from individual artifacts. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions. If you update the list of default run permissions, the change affects new artifacts only.

1. If needed, open the Admin Console (click in the MashZone NextGen Hub main menu).
2. Expand the **Security & Policies** tab and click **Default Permissions**.
3. Clear or set the **Users** or **Groups** options as needed. Enter part of a user or group name and click **Search**.

   A list of users and/or groups that match your criteria displays in the left pane.

   Use `MashZone NextGen` as the search term and search for groups to get a list of the built-in MashZone NextGen groups.
4. Drag a group or user into one of these buckets:
   - **Default principals who can run Services** grants run permissions to all new mashables.
   - **Default principals who can run Mashups** grants run permissions to all new mashups.
   - **Default principals who can run Apps** grants run permissions to all new apps (basic, custom or workspace).
5. Click **X** to delete a user or group from the list of default permissions in any of these buckets.
6. Click **Save these changes**.

Set View Permissions with a Search Filter

View permissions are defined in the built-in MashZone NextGen groups that you assign to users. View permissions determine what artifacts appear in search results in MashZone NextGen Hub and the AppDepot and the activity feed in the MashZone NextGen Hub home page.

By default, any authenticated user can see any app in search results in the AppDepot. Users that have permission to work in the MashZone NextGen Hub can also see any mashable, mashup or app in search results in MashZone NextGen Hub.

With this default, users can find any artifact and can open the artifact page for any artifact, even if they are not permitted to use that artifact. If they do not have run permission for an artifact that they open, MashZone NextGen displays an error message. Users must request run permissions for the artifact from a MashZone NextGen administrator or the artifact's owner.

This design encourages discovery and reuse of artifacts, leveraging existing assets throughout your organization. You can make the default view permissions more restrictive to allow search results to include only those artifacts that a user is permitted to run.

To set view filters

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Security section and click **Search Filters**.
3. Change the filter as needed:
   - **Show all items**: this is the default search filter that allows users to see any artifact in search results.
   - **Show only viewable items**: this option is reserved for future use. Currently, it also includes all artifacts in search results.
   - **Show only executable items**: set this option to limit search results to those artifacts that a user also has permission to run.
4. Click **Save settings**.

Enable or Disable Authorization

Authorization is enabled by default in MashZone NextGen. You can disable authorization checks to simplify workflow during development.

To enable/disable authorization

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Securities & Policies** section.

3. Click **Permissions** and set or clear the **Enable authorization** property.
4. Click **Save permission settings**.

MashZone NextGen Server Configuration

Basic Configuration and Logging

- "Manage Licenses for MashZone NextGen, Universal Messaging and BigMemory" on page 1572
- "Support International Character Sets and Locales" on page 1631
- "Change the MashZone NextGen HubTheme" on page 1634
- "Configure the MashZone NextGen Server with Custom Ports" on page 1638
- "Configure the MashZone NextGen Server to Work with a Proxy Server" on page 1640
- "Define a Proxy Server Whitelist for MashZone NextGen" on page 1641
- "Configure MashZone NextGen for SSL and Digital Certificates" on page 1643
- Configuring "MashZone NextGen Notifications" on page 1653
- Configuring "MashZone NextGen Logging" on page 1650 and viewing logs
- "Manage Data Sources and Drivers" on page 1679
- "Manage Categories for Mashups, Mashables and Apps" on page 1686
- "Manage Providers for Mashups, Mashables and Apps" on page 1686
- "Work With MashZone NextGen Attributes" on page 1687

See also "MashZone NextGen and MashZone Repositories" on page 1737 for links to configuration for the MashZone NextGen Repository.

Mashables, Mashups and Wires Configuration

- "Manage Artifact Attributes" on page 1690
- "Manage Data Sources and Drivers" on page 1679
- "Configure the Default Operations Generated for Database Mashable" on page 1684
- "Disable Mashup Features" on page 1691
- "Configure Mashable HTTP Request Timeouts" on page 1693
- "Configure HTTP Response Header Forwarding" on page 1692
- "Enable or Disable the Snapshot Feature" on page 1693
- "Set Web Feed Normalization" on page 1694

- "Handle SOAP Encoding Errors for WSDL Services" on page 1694
- "Add XML Schemas to the Wires Mapper Block" on page 1695

Add-Ons, Performance, Deployment and Miscellaneous

- "BigMemory for Caching, Connections and MashZone NextGen Analytics" on page 1661
- "Configure MashZone NextGen Connections to SharePoint" on page 1655
- "Memory Configuration for the MashZone NextGen Server" on page 1627
- "Deploying MashZone NextGen Instances, Clusters or Artifacts" on page 1748
- See also "MashZone NextGen Security" on page 1602 for links to server configuration tasks for authorization.

Memory Configuration for the MashZone NextGen Server

MashZone NextGen is initially installed with default memory settings for a small web application. Your actual memory requirements may vary significantly based on your expected load, throughput and environment.

Note: With release 3.7, MashZone NextGen is no longer supported on 32-bit architectures which have memory access limitations from Java.

If you are working with large datasets in MashZone NextGen Analytics or you are deploying MashZone NextGen in a staging or production environment, you may need to tune this default memory configuration. Which options you can configure depends on your usage and environment considerations:

If	See
■ Your computer has less than 4G of RAM memory, or ■ You have *not* installed BigMemory Server(s).	"Configuration When MashZone NextGen Uses Only Heap Memory" on page 1627
■ You have installed BigMemory Server(s), or ■ Your computer has more than 4G of RAM memory.	"Configuration When MashZone NextGen Uses Heap and Off-Heap Memory" on page 1629

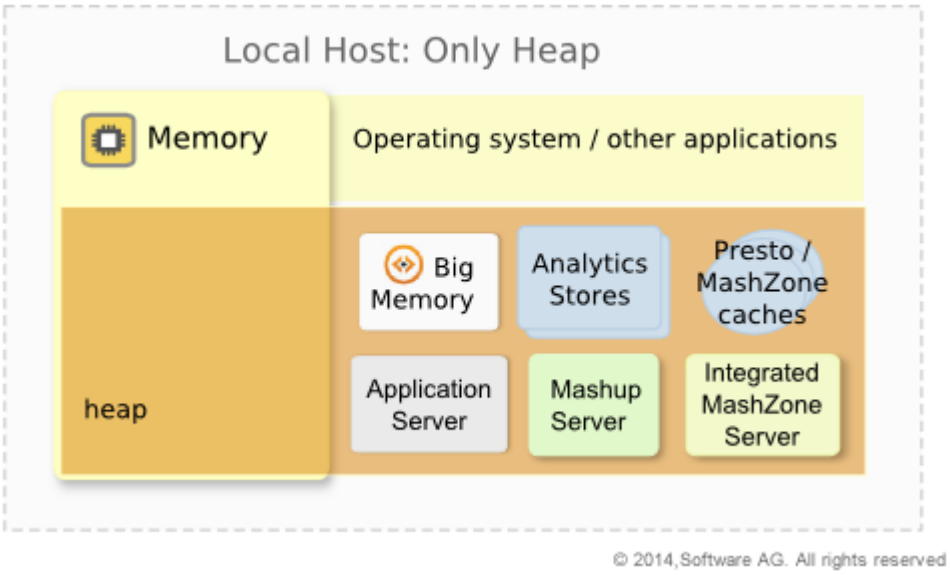
Configuration When MashZone NextGen Uses Only Heap Memory

The initial default Java heap memory settings for MashZone NextGen are appropriate for small applications or development environments 1G as the maximum heap size.

Both the Integrated MashZone Server and the Event Service that are deployed with MashZone NextGen also use this heap space.

Note: If you are using MashZone NextGen Analytics with large datasets and limited memory (less than the recommended 4G minimum), configuring BigMemory to use off-heap memory is *not* recommended as it can adversely affect performance.

Some portion of available memory must be reserved for the operating system and any other applications on this host, as shown in the following figure:



How much memory to allocate to MashZone NextGen depends on available memory, requirements for the Integrated MashZone Server and Event Service and what other applications may run on this computer.

Note: For more information and suggestions on memory requirements for the Integrated MashZone Server and Event Service, see "Tune Memory/Caching for the Integrated MashZone Server" on page 1697.

To update memory configuration

1. In a text editor of your choice, open the application server configuration file appropriate for your operating system:
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat, for Windows
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh, for Linux, OS/X or UNIX
2. Change any of these Java memory options:

-Xms	Default = `512M`

-Xmx	Default = `1G`
-XX:PermSize	Default = `128M`
-XX:MaxPermSize	Default = `256M`

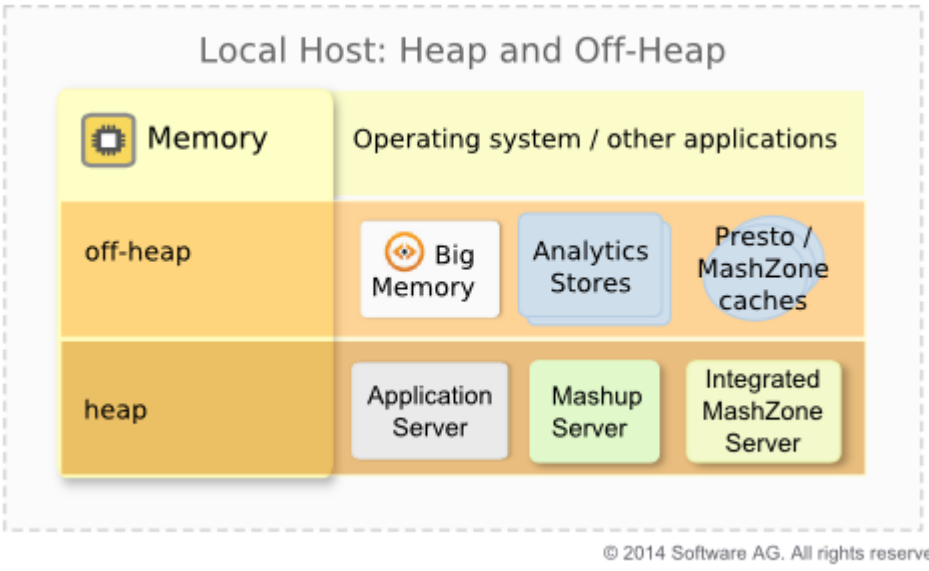
3. Save your changes and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Configuration When MashZone NextGen Uses Heap and Off-Heap Memory

MashZone NextGen should be configured to use both heap and off-heap memory only when the available memory supports this adequately and you have also installed BigMemory Servers.

Note: You must have installed a copy of your BigMemory license in MashZone NextGen to use off-heap memory. See "Manage Licenses for MashZone NextGen, Universal Messaging and BigMemory" on page 1572 for instructions.

With combination heap and off-heap memory, as this figure shows, BigMemory uses off-heap memory for the MashZone NextGen Analytics In-Memory Stores and MashZone NextGen caches. All other MashZone NextGen processing, including the Integrated MashZone Server and Event Service that are deployed with MashZone NextGen, remains in heap.



The total available off-heap memory may be limited to local off-heap memory as shown above, or it may include additional off-heap memory on external hosts if you have installed BigMemory Server arrays.

Note: For more information and suggestions on memory requirements for the Integrated MashZone Server and Event Service, see "Tune Memory/Caching for the Integrated MashZone Server" on page 1697.

To update memory configuration:

1. In a text editor of your choice, open the application server configuration file appropriate for your operating system:
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat, for Windows
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh, for Linux, OS/X or UNIX
2. Change or add either of these memory options used with BigMemory:

-Dpresto.bm.maxOffHeap	Default = `1G` This is the maximum size of *local* off-heap memory that BigMemory can use for the MashZone NextGen Analytics In-Memory Stores and MashZone NextGen caches. This property sets off-heap memory limits in the *MashZoneNG-config* /ehcache.xml configuration file. The total size of off-heap memory may include additional, external memory depending on how BigMemory is deployed.
-XX:MaxDirectMemorySize	Default = `1500M` This Java memory option must be set to allow access to both off-heap and an additional allocation for Java. The value of this option must *always* be larger than the memory allocated to off-heap. A good rule of thumb is at least 500M more.

3. Change or set any of these Java memory options:

-Xms	Default = `512M`
-Xmx	Default = `1G` See the Java Tuning White Paper for more information and suggestions.
-XX:PermSize	Default = `128M`
-XX:MaxPermSize	Default = `256M`

4. Save your changes and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Support International Character Sets and Locales

International character sets are the alphabets, characters, glyphs and other symbols used in any non-English language. Technically, this includes any non-ASCII characters. To handle these characters properly, MashZone NextGen must know the character set and how it is digitally represented - the *character encoding*.

Note: MashZone NextGen uses the UTF-8 character encoding to handle character sets for all languages. Both UTF-8 and UTF-16 can represent any Unicode character.

Unicode defines a unique encoding for every character in world languages that are currently in active use as well as some well-known dead languages, such as ancient Greek.

Locale identifies the language used and potentially specific regional spelling or usage aspects of the language, such as differences between American English (EN_us) versus Australian English (EN_au). Locale also identifies the formats used to present dates, times and numbers for that region.

Both the character encoding and locale help to ensure that MashZone NextGen properly handles and presents data to users. The areas of configuration involved in this support include:

- *MashZone NextGen Repository*: the character encoding for this repository determines what character sets users can use when they create artifacts. The timezone for the MashZone NextGen Repository also affects timestamps shown in MashZone NextGen Hub.

  For configuration options and instructions, see " MashZone NextGen Repository Encoding and Timezone" on page 1632.

- *Artifact data*: MashZone NextGen needs to know the character encoding of mashable responses to properly handle this data as well as mashup results and the data shown in apps. This defaults to UTF-8 when users register mashables, but you can provide a mechanism so that users can override this default.

  The locale also affects how data is displayed. This also has a default that can be overridden.

  For configuration options and instructions, see "Mashable, Mashup and App Encodings and Locales" on page 1633.

- *Display options*: in most cases, date, time and numeric data are shown to users based on browser settings or a default locale. Some views allow users to choose date and time formats.

  See "Date, Time and Numeric Display Options" on page 1633 for details.

- *Logging*: you can also support international characters and different locales for the messages sent to MashZone NextGen logs. See "Message Log and Default Locales" on page 1634 for details.

MashZone NextGen Repository Encoding and Timezone

Set the Repository Character Encoding

The character encoding for the MashZone NextGen Repository is defined when you create the database that will host the repository. To support international character sets, this should be set to UTF-8, or for some databases UTF-16.

Important: Because of known issues, artifact names and the IDs that are generated from these names are restricted to ASCII characters. The syntax and encoding names you must use are specific to each database. For more information, see:

- Documentation for your database
- And either:
  - Move MashZone NextGen and MashZone Repositories to Microsoft SQL Server
  - Move the MashZone NextGen and MashZone Repositories to MySQL
  - Move the MashZone NextGen and MashZone Repositories to Oracle
  - Move the MashZone NextGen and MashZone Repositories to PostGres

Set the Repository Timezone or Offset

The default timezone that is used to record timestamps such as the created date and time for an artifact is the timezone for the host of the MashZone NextGen Repository. You can change the timezone used to save repository timestamps or set an offset so that repository timestamps are displayed in a different timezone:

- Force the timezone that the MashZone NextGen Repository uses to match the timezone for the MashZone NextGen Server. See "Synchronize the MashZone NextGen Repository and MashZone NextGen Server Time Zones" on page 1740 for instructions.
- Configure the display timezone in MashZone NextGen Hub as an offset from UTC. Note that this does not affect the actual timezone recorded in the MashZone NextGen Repository.

  This offset is defined in the `repositoryTimezoneOffset` property in *web-apps-home* /presto/hub/config.js file. This property is undefined by default.

  Edit this JSON property, setting the number of minutes as a UTC offset. For example, `300` sets this to Eastern Standard Time while `-180` sets this to Arabic Standard Time.

  Once the property is saved, restart the MashZone NextGen Server.

Mashable, Mashup and App Encodings and Locales

Default Mashable and Mashup Encoding and Overrides

Some types of mashables may include character encoding information in the response, such as XML files, while others do not, such as CSV. MashZone NextGen uses UTF-8 as the default character encoding for mashable and mashup results.

For mashables that are file-based, MashZone NextGen uses the Java system property `file-encoding` to ensure this default is available for mashables that cannot specify an encoding, such as CSV files. This Java property is set to default to UTF-8 in startup scripts for the application server (*MashZoneNG-install* /apache-tomee-jaxrs/bin/setup.[bat|sh]).

You can provide a mechanism to override this default encoding for individual mashables by defining an artifact attribute named `encoding_override`. See "Setting a Character Encoding for a Mashable" on page 182 for information on how to make this override available.

Mashable, Mashup or App Locales

When users register mashables, the default locale for the mashable is set to the locale from the user's browser, if that locale is available and is a supported locale for MashZone NextGen. If the user's locale is not available or it is not a supported locale, MashZone NextGen uses the locale from the JVM for the MashZone NextGen Server as the default locale.

Users can override this default locale for mashables or set the locale for mashups and apps with the **Region** field. See "Change an Artifact's Locale" on page 92 for instructions.

You can also set the default locale for the JVM for MashZone NextGen. See "Message Log and Default Locales" on page 1634 for instructions.

Date, Time and Numeric Display Options

In general, MashZone NextGen displays dates, times and numeric data using the formats defined by the browser's locale for the current user. If no locale information is available from the browser, MashZone NextGen use the default system locale which typically is EN_us.

Some built-in views in MashZone NextGen, allow users to choose a date and time pattern for result data such as `mm/yyyy`, for the month and year, or `EEE MMM dd, yyyy`, for the day of the week, month name, day and year. This pattern determines the components of dates or times that display in that view, but the language, order and delimiters used in the display are determined by the user's locale or the default locale.

Message Log and Default Locales

MashZone NextGen uses the default locale defined for the JVM for all messages that are added to MashZone NextGen logs. These defaults may also be used as the locale for artifacts if no locale is defined by users or provided by the client browser.

To set the locale for the JVM for MashZone NextGen using Java properties

1. In a text editor of your choice, open the setenv.bat file, for Windows systems, or the setenv.sh file, for Linux, OS/X or UNIX systems, in the *MashZoneNG-install* /apache-tomee-jaxrs/bin folder.
2. Add a line just below the initial lines which set the `JAVA_OPTS` environmental variable.
   - For Windows, add:

     `set JAVA_OPTS=%JAVA_OPTS% -Duser.county=`*`country-code`*`-Duser.language=`*`language-code`*
   - For Linux, OS/X or UNIX, add:

     `JAVA_OPTS="$JAVA_OPTS -Duser.county=`*`country-code`*`-Duser.language=`*`language-code`*`"`

   For example:

```
JAVA_OPTS="-XX:PermSize=128M -XX:MaxPermSize=256M  -Xms512m  -Xmx1G"
JAVA_OPTS="$JAVA_OPTS -DPRESTO_HOME=$CATALINA_BASE"
JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8"
JAVA_OPTS="$JAVA_OPTS -Dpresto.raql.udf.libsReload=false
  -Dpresto.raql.udf.libsDir=$CATALINA_BASE/../raql-udfs"
JAVA_OPTS="$JAVA_OPTS
  -Dlog4j.configuration=file://$CATALINA_BASE/conf/log4j.properties"
JAVA_OPTS="$JAVA_OPTS -Duser.country=CA -Duser.language=fr"
# (this is set by the installer)
JAVA_HOME=${JAVA_HOME}
...
```

3. Save your changes to the file.
4. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Change the MashZone NextGen HubTheme

You can change the following aspects of the main MashZone NextGen Hub menu:

- The logo.

  This image should be either GIF, PNG or JPG. The suggested size is 150 pixels wide and 40 pixels high.
- The favorites icon (favicon) that appears in browser tabs or address bars.

This image should be GIF, PNG or JPG and must have a `.ico` file extensions. The size must be 16 pixels square.

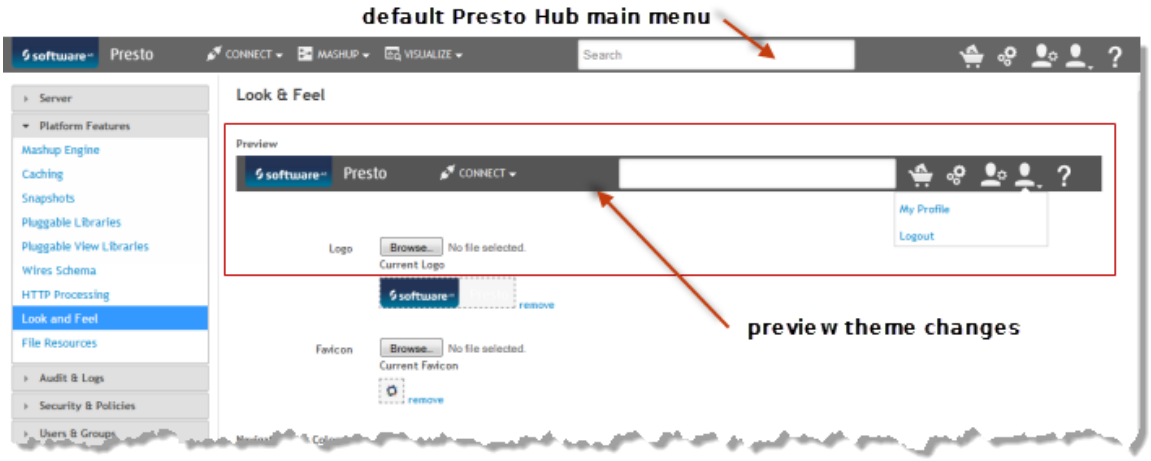
- The color of the main menu background. This can be a single color, for a flat look, or two colors used in a gradient for a curved look. The default menu color is a flat black.
- The color to use for the text of main menu items.

  By default, the text is black against a white background (unselected) or white (selected) against the background color for the main menu. If you change the menu item text color, this single color is used against both backgrounds.

To change the main menu theme

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Open the Platform section and click **Look and Feel**.

   The Look and Feel page shows a preview of the current menu settings with the properties you can change:



3. To change the logo or favicon:
   a. Click **Browse**.
   b. Find and select the image you want to use.
   c. Once the image has been uploaded, refresh your browser to see this logo in the MashZone NextGen Hub menu.

   To remove a custom logo and return to the default MashZone NextGen logo, click **Remove** next to the current logo image in the page. The browser automatically refreshes.
4. To change the background color for the menu and use a flat look:
   a. Use the color picker for the Top Gradient color and select the color you want.
   b. Choose the same color for the Bottom Gradient color.

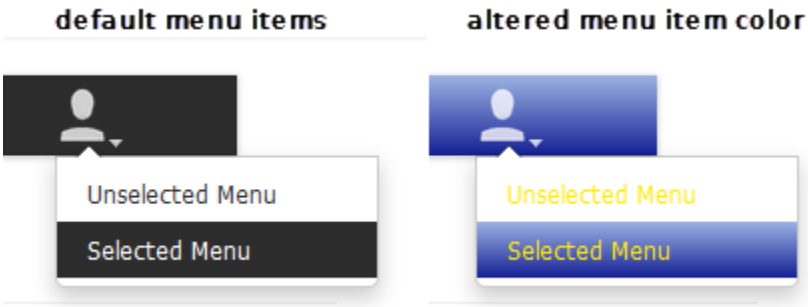
The menu preview updates showing you what this will look like:

5. To change the background color for the menu and use a curved look, use a gradient:
   a. Use the color picker for the Top Gradient color and select the color you want. Typically this is a lighter color than the bottom gradient.
   b. Choose a different color for the Bottom Gradient color. Typically this is a darker color than the top gradient.

   The menu preview updates showing you what this will look like:

6. To change the color of menu items, use the color picker for **Menu font color**.

   This single color is used for both selected and unselected item so it is better to choose a color that is easily visible against both white and the menu background color(s).



7. Click **Save Changes** to apply the new theme.

To change the theme of the MashZone NextGen Dashboard component please have a look at "Edit style templates" on page 196 of MashZone NextGen Dashboard.

Set the default chart theme

In Admin Console you can set a predefined chart theme as the default global chart theme.

The selected chart theme is then preselected by default in the theme selector in the chart and app wizard of every chart. If required, users can still change the preselected chart theme in the theme selector.

To set the default chart theme

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Platform Features** to expand the feature section and click **Look and Feel**.
3. Click **Charts** to open the charts tab.
4. Choose a **Default Chart Theme** in the drop-down menu.
5. Click **Save Changes** to apply the selected theme.

The selected theme is set as the global default chart theme.

Edit style templates

You have MashZone NextGen administrator permissions.

You can edit the style templates supplied with MashZone NextGenDashboard. Editing the style templates enables you to customize the look and feel of your dashboards and the Dashboard component itself, e. g., colors schemes, fonts, brand logo or background colors.

The application style template file application.less is located in the following folder on the MashZone NextGen server.

<*MashZone NextGen installation*> \apache-tomee-jaxrs\webapps\presto\hub\dashboard \assets\custom-look-and-feel\application.

The dashboard style template files are located in the following folder on the MashZone NextGen server. The default dashboard template file is named default.less.

<*MashZone NextGen installation*> \apache-tomee-jaxrs\webapps\presto\hub\dashboard \assets\custom-look-and-feel\dashboard

The style templates provide basic variables that are used as default by many components, e.g. @header-color. You can override the basic variables by defining more specific, customized variables, e.g. @header-dashboard-name-color.

For example, if @header-color is black and @header-dashboard-name-color is red and both variables are the only ones defined, all labels in the header will be black except of the dashboard name label, it will be red.

All available application and dashboard variables are described in the corresponding style template files.

Procedure

1. Open the relevant template style file in your text editor.
2. Edit the style parameters according your requirements.
3. Save your changes.
4. Reload the changed style template files into MashZone NextGen Dashboard.
   a. Open MashZone NextGen Dashboard
   b. Click **Manage** > **Change style template** in the Dashboard main menu.
   c. To reload the application style template click **Compile to CSS**.
   d. To reload the dashboard style template select the relevant dashboard style template in the drop-down menu and click **Compile selected to CSS**.
5. Click **OK**.

The changed style template has been reloaded into MashZone NextGenDashboard and the style templates apply to the application or dashboard.

Configure the MashZone NextGen Server with Custom Ports

Port configuration is initially set when you install MashZone NextGen. If you change these ports or you need to host multipleMashZone NextGen Servers on one host, you must update configuration in theMashZone NextGen Server. You may also need to change ports for the MashZone NextGen Repository and for TomEE.

Change MashZone NextGen Server Ports

The host name and port for theMashZone NextGen Server defaults to localhost and 8080 respectively. The port is typically defined in configuration for TomEE, the application server that hosts MashZone NextGen.

To change the host and port information, you must:

1. Update port configuration for the application server that hostsMashZone NextGen in *MashZoneNG-install* /apache-tomee-jaxrs/conf/server.xml in the `<Connector>` elements for HTTP and/or HTTPS.
2. Update host and port information used for user email notifications.

   Note: This configuration is used to generate full, correct links in notifications that are sent to users via email. It does not affect the MashZone NextGen Server. You can also update this information if you use a load balancer so that generated links point to the load balancer.

   To change this configuration:

a. Click to open the Admin Console.

b. In the Server section, select **Server Host and Port**.

c. Update **Host** and **Port**.

d. Click **Save server information**.

3. Restart the MashZone NextGen Server.See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Change MashZone NextGen Repository Ports

If you are running multiple instances of the MashZone NextGen Server in one host with separate instances of the MashZone NextGen Repository, you may need to use different ports for each database instance:

The default MashZone NextGen Repository	No updates to ports are needed as the Derby database is embedded.
The MashZone NextGen Repositoryis hosted in a robust database	You must use different ports for each MashZone NextGen Repository instance.

To update MashZone NextGen Repository ports in the Admin Console.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **MashZone NextGen Repositories** section
3. Select **Metadata Repository** and:
   a. Change the **JDBC URL** to the correct host and port number.
   b. Click **Save**.
4. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

TomEE Application Server Port

If you are running multiple instances of the MashZone NextGen Server in one host, you must have separate application server instances for each. You must update the following ports:

Configuration File	Element
MashZoneNG-install /apache-tomee-jaxrs/conf/server.xml	`<Server>` to set the command port

Configuration File	Element
	`<Connector>` for the HTTP port
	`<Connector>` for the HTTPS port

Configure the MashZone NextGen Server to Work with a Proxy Server

MashZone NextGen is *only* compatible with HTTP proxy servers.

If you have a proxy server in your environment that MashZone NextGen should use, you *must* add configuration information to the MashZone NextGen Server for the proxy server. You can also define a *whitelist* of addresses that do not require proxy server access. See "Define a Proxy Server Whitelist for MashZone NextGen" on page 1641 for more information.

To configure a proxy server

1. Start the MashZone NextGen Server.

   See "Start the MashZone NextGen Server" on page 1570 for instructions.
2. Open MashZone NextGen Hub at `http://`*appl-server:port* `/presto`.
3. Enter your **Username** and **Password** and click Login.

   Initially, you can use the default administration account:

   - *username* = `Administrator`
   - *password* = `manage`
4. Click Admin Console in the MashZone NextGen Hub main menu.
5. In the Server section, click **Proxy Settings**.
6. Set the **Enable Proxy** option.
7. Set the following connection properties for your proxy server:
   - *Host* = the host name or IP address for the proxy server. This is required.
   - *Port* = the port number for the proxy server. This is required.
   - *Username* = the user name that the MashZone NextGen Server should use to connect to the proxy server. This is only required if your proxy server requires credentials.
   - *Password* = the password that the MashZone NextGen Server should use to connect to the proxy server. This is only required if your proxy server requires credentials.

8. If needed, define a whitelist of addresses that should *not* use the proxy server. See "Define a Proxy Server Whitelist for MashZone NextGen" on page 1641 for instructions.
9. Click **Save proxy settings**.

Define a Proxy Server Whitelist for MashZone NextGen

If you have configured a proxy server for the MashZone NextGen Server, you can define a whitelist of domains, hosts or IP addresses that do *not* require access through the proxy server.

To define a proxy server whitelist

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. In the Server section, click **Proxy Settings**.
3. Set the following properties:
   - *Bypass IP List* = enter one or more IP address, separated by commas, that the MashZone NextGen Server should access without the proxy server. To use wildcards in IP addresses, see "Using Regular Expressions in a Whitelist" on page 1641.
   - *Bypass Host List* = enter one or more fully-qualified host names, separated by commas, that the MashZone NextGen Server should access without the proxy server.To use wildcards in IP addresses, see "Using Regular Expressions in a Whitelist" on page 1641.
   - *Bypass Domain List* = enter one or more domain names, separated by commas, that the MashZone NextGen Server should access without the proxy server. To use wildcards in IP addresses, see "Using Regular Expressions in a Whitelist" on page 1641.
4. Click **Save proxy settings**.
5. Log out of the MashZone NextGen Hub.
6. Stop and restart the MashZone NextGen Server to apply these changes.

Using Regular Expressions in a Whitelist

All of the whitelist properties accept *regular expressions* to define sets of IP, host or domain name addresses using wildcards. You can use any valid regular expression character, however, the most common wildcard characters that you may use are:

Replace Any Single Character	.
Replace Any Number of Characters	.*

Examples and solutions for the most common patterns include:

- Specifying Literal Dot Separators
- Specifying Domains
- Specifying Host Names

Specifying Literal Dot Separators

Because the dot character is used as a wildcard in regular expressions and is also the standard separator for groups in IP addresses, domain names and host names, you can get unintended results when using wildcards. For example, this is a valid regular expression for IP addresses:

```
139.16.1.*
```

On Windows systems, many administrators would expect this to expression to "match any IP address with first-through-third groups of 139, 16 and 1 respectively" such as `139.16.1.10` and `139.16.1.35`.

This would actually match either of these IP addresses:

```
139.16.1.10
```

```
139.16.11.120
```

In most cases, the difference between a literal dot and the dot as a wildcard character doesn't make a difference. If you need to clarify a whitelist entry to match a literal dot, use `\.` instead.

The expression `139\.16\.1\..*` would correctly match `139.16.1.10` and `139.16.1.35` but would not match `139.16.11.120`. In many cases, you could also simplify this to `139.16.1\..*` to get the correct behavior.

Specifying Domains

With domains, you must specify a wildcard at the beginning of the domain name. This example is not a valid domain name expression:

```
mydomain.com
```

This entry would *not* match a host name of `east.mydomain.com` or `east.customers.mydomain.com`. To specify the domain correctly, enter:

```
.*mydomain.com
```

Specifying Host Names

In whitelist properties, host names are fully-qualified. Thus `stives` is not a valid host name while `stives.customers.mydomain.com` is valid. A host name expression of `.*customers.mydomain.com` would match all of these hosts:

```
stives.customers.mydomain.com
```

```
cour.customers.mydomain.com
```

`tempcustomers.mydomain.com`

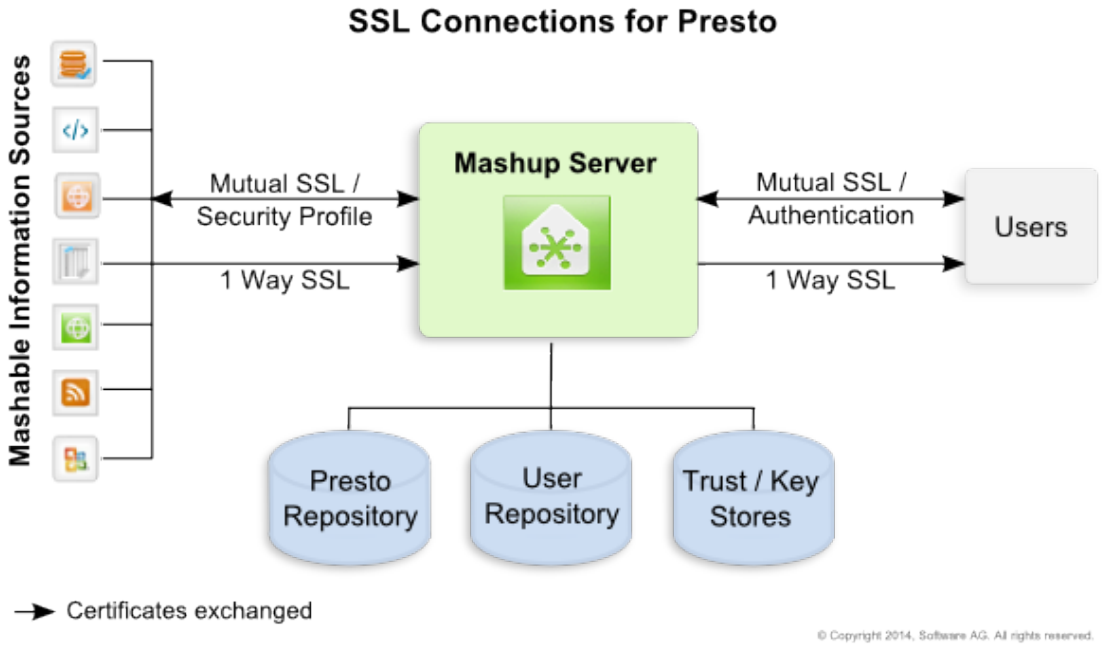
Note that an expression of `.*.customers.mydomain.com` would also match these same three hosts.

You may need to specify literal dot separators in host names also to properly clarify the expressions. If in this example you did *not* want `tempcustomers.mydomain.com` to be matched, you would need an expression such as `.*\.customers.mydomain.com`. See "Specifying Literal Dot Separators" on page 1642 for more information.

Configure MashZone NextGen for SSL and Digital Certificates

MashZone NextGen expects HTTP as the default transport protocol from clients to the MashZone NextGen Server. Connections from the MashZone NextGen Server to mashable information sources typically also use HTTP.

MashZone NextGen supports HTTPS and SSL for connections from clients or connections to many types of mashables, such as web feeds or REST and WSDL web services, or through direct connections using EMML. MashZone NextGen can also use digital certificates from clients in user authentication.



The certificate store, certificates and configuration needed to support SSL in MashZone NextGen depends on the connection requirements, as shown below:

	Certificate Store and Certificates		Store Configuration			MashZone NextGen Configuration	
	Key	Trust	Java	App Server	MashZo NextGen	Authentication	Security Profiles
"One-Way SSL to MashZone NextGen" on page 1647.	✓			✓			
Mutual to MashZone NextGen See "Configure Mutual SSL Between Users and MashZone NextGen" on page 1646.	✓	✓		✓		✓	
"One-Way SSL to Mashable Information Sources" on page 1647.		✓	can be in either				
"One-Way SSL to Information Sources Using <directinvoke> in Mashups" on page 1648.		✓	✓				

	Certificate Store and Certificates		Store Configuration			MashZone NextGen Configuration	
	Key	Trust	Java	App Server	MashZo NextGen	Authentication	Security Profiles
Mutual to Mashables See "Configure Mutual SSL Between MashZone NextGen and Mashable Information Sources" on page 1646	✓	✓			✓		✓

See also "The Certificate Store and Certificates" on page 1645 for more information:

The Certificate Store and Certificates

Both key stores and trust stores are *certificate stores* to store and manage the *key certificate pairs* or *public certificates* used in secure connections with the SSL protocol. Key stores manage key certificate pairs and trust stores manage the public certificates of trusted peers.

Key Certificate Pairs

For MashZone NextGen, the key certificate pair stored in the key store identifies the MashZone NextGen Server to users, for both one-way and mutual SSL. The key certificate pair identifies the MashZone NextGen Server to mashable information sources for mutual SSL.

You must generate a key certificate pair for MashZone NextGen. Typically you also have the key certificate pair signed by a Certificate Authority and import this into the certificate store using the Java `keytool` utility or other certificate management tools.

Trusted Peer Cerficates

The public certificates from peers are stored in the trust store and identify users, for mutual SSL, or identify information sources (mashable or direct sources used in mashups), for one-way or mutual SSL.

When public certificates for peers are signed by well known Certificate Authorities, they are automatically verified and imported into the trust store. If public certificates are self-

signed or signed by an unknown Certificate Authority (the CA root certificate is not found in the trust store), you must obtain and import the public certificates to the trust store before the first connection occurs during:

- User login.
- Mashable registration.
- Direct invocation in mashups.

The Certificate Store

You can use a single certificate store as both the key store and trust store for MashZone NextGen or you can use separate certificate stores. You can use an existing certificate store for MashZone NextGen, such as the default certificate store shipped with some application servers. Or you can create a new certificate store using the Java `keytool` utility.

See Java keytool documentation for more information, commands and instructions on managing key certificate pairs, trusted certificates and certificate stores.

Configure Mutual SSL Between Users and MashZone NextGen

The MashZone NextGen Server and users both exchange certificates. MashZone NextGen can also be configured to use user digital certificates for authentication. The connection requires:

- Store and Certificates:
  - A certificate store as key store and trust store for the MashZone NextGen Server.
  - A key certificate pair for the MashZone NextGen Server.
  - Public certificates in the trust store for any user public certificates that are self-signed.

    You must add self-signed certificates to the trust store before these users login. See "Trusted Peer Cerficates" on page 1645 for more information.

  See "The Certificate Store and Certificates" on page 1645 for more information.
- Configuration in the application server hosting MashZone NextGen to use the HTTPS port. This also includes configuration identifying the key store and trust store for the MashZone NextGen Server. See "Configure HTTPS and Certficate Stores in the Application Server" on page 1648 for instructions.
- Optional configuration in MashZone NextGen to use digital certificates for user authentication. See "Authentication with Digital Certificates/SSL" on page 1614 for instructions.

Configure Mutual SSL Between MashZone NextGen and Mashable Information Sources

Both the information source and MashZone NextGen exchange certificates.

Note: For mashups, you must use the <invoke> statement to connect to information sources that require mutual SSL. The <directinvoke> statement in EMML only supports one-way SSL connections.

This scenario uses the SSL security profile that is provided in MashZone NextGen. It requires:

- Store and Certificates:
  - A certificate store as key store and trust store for the MashZone NextGen Server.
  - A key certificate pair for the MashZone NextGen Server.
  - Public certificates in the trust store for any information sources that have self-signed certificates.

    You must add self-signed certificates to the trust store before the mashable information source can be registered. See "Trusted Peer Cerficates" on page 1645 for more information.

  See "The Certificate Store and Certificates" on page 1645 for instructions.
- Configuration in MashZone NextGen for both the key store and trust store. See "Configure Certificate Stores in MashZone NextGen" on page 1649 for instructions.
- Security Profile configuration for each mashable information source. You provide this configuration when you register the mashable. See "Mashable Authentication with Security Profiles" on page 161 for more information.

One-Way SSL to MashZone NextGen

This requires:

- A key store and a key certificate pair for MashZone NextGen. See "The Certificate Store and Certificates" on page 1645 for more information.
- Configuration in your application server for the HTTPS port to MashZone NextGen and the key store. See "Configure HTTPS and Certficate Stores in the Application Server" on page 1648 for instructions.

One-Way SSL to Mashable Information Sources

This requires:

- A trust store for MashZone NextGen. See "The Certificate Store and Certificates" on page 1645 for more information.
- Configuration for the trust store in either:
  - The application server hosting the MashZone NextGen Server. See "Configure HTTPS and Certficate Stores in the Application Server" on page 1648 for instructions.
  - Java. See "Update SSL Configuration for Java" on page 1649 for instructions.

- Self-signed certificates, if any, for mashable information source using one-way SSL. You must add these certificates to the trust store before the mashable information source can be registered. See "Trusted Peer Cerficates" on page 1645 for more information.

One-Way SSL to Information Sources Using <directinvoke> in Mashups

This requires:

- A trust store for MashZone NextGen. See "The Certificate Store and Certificates" on page 1645 for more information.
- Configuration for the trust store in Java. See "Update SSL Configuration for Java" on page 1649 for instructions.

  Note: EMML uses the certificate stores defined in Java.

- Self-signed certificates, if any, for the information source using one-way SSL. You must add these certificates to the trust store before the mashup invokes these information sources. See "Trusted Peer Cerficates" on page 1645 for more information.

Configure HTTPS and Certficate Stores in the Application Server

Configuration for SSL for MashZone NextGen can be defined in the application server that hosts the MashZone NextGen Server. These instructions discuss the basic steps for configuring SSL in Tomcat. See Tomcat Documentation or the documentation for your application server for detailed information.

1. If you do not yet have a key store, trust store and certificate for the MashZone NextGen Server, find or create these stores and certificate. See "The Certificate Store and Certificates" on page 1645 for instructions.
2. Configure Tomcat for secure connections from clients to the MashZone NextGen Server:
   a. Edit the server.xml file for Tomcat to uncomment and configure the <Connector> element for SSL/HTTPS 1.1. For example:

```
<Connector port="8443" protocol="HTTP/1.1"
  SSLEnabled="true" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25"
  maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" scheme="https" secure="true"
  clientAuth="true" sslProtocol="TLS"
  keystoreFile="conf/tomcat.jks"
  keystorePass="keystrpwd"
  truststoreFile="conf/tomcat.jks"
  truststorePass="truststrpwd" />
```

   This example uses the default Tomcat port, 8443, and mutual SSL, based on the `clientAuth` value. If this was a one-way connection, you would set `clientAuth` to false. This example also uses the default Tomcat certificate store, `conf/`

`tomcat.jks`, as both the key store and the trust store. See Tomcat documentation for information on other properties.

b. Once you have configured an HTTPS port in your application server, update port configuration for the MashZone NextGen Server to listen to that port. See "Configure the MashZone NextGen Server with Custom Ports" on page 1638 for more information on this step.

3. If needed, enable MashZone NextGen authentication to use certificates. See "Authentication with Digital Certificates/SSL" on page 1614 for instructions.

Configure Certificate Stores in MashZone NextGen

For secure connections to mashable information sources using mutual SSL and SSL security profiles, you must add key store and trust store configuration to MashZone NextGen in the Admin Console:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Security** tab and click **Certificates**.
3. Enter the **URL** and **Password** for the certificate store to use as the key store.
4. Click the **Truststore** tab and enter the **URL** and **Password** for the certificate store to use as the trust store.

   In most cases, this will be the same certificate store as the key store
5. Click **Save settings**.

   A list of all key certificates (identifying this MashZone NextGen Server or other MashZone NextGen Servers) in the key store displays in the Keystore tab.

   A list of all trusted certificates (public certificates for trusted information sources) displays in the Truststore tab.

Update SSL Configuration for Java

With the EMML <directinvoke> statement, certificates for secure endpoints are validated against the default trust store for Java (the JRE). One-way SSL for mashable information sources may also use the default trust store for Java.

Note: See http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html#CustomizingStores for more information on the default JRE trust store.

Initially, this may not be the trust store you have configured for the MashZone NextGen Server in the application server and/or the Admin Console. This can cause security errors for <directinvoke> statements or mashable information sources.

To avoid these errors, you can configure the JRE to use the trust store for the MashZone NextGen Server:

1. Open the appropriate startup script (below) for the application server hosting the MashZone NextGen Server in any text editor:
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat file for Windows environments
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX environments
2. Add these system properties in the Java options in this script:
   - `-Djavax.net.ssl.trustStore=`*`/path/to/mashup-server/truststore`*: this is the absolute path to the trust store for the MashZone NextGen Server.
   - `-Djavax.net.ssl.trustStorePassword=`*`truststore-password`*: this is only required if the MashZone NextGen Server's trust store uses a password.
3. Save your changes to the script and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

MashZone NextGen Logging

In addition to logging from your application server, MashZone NextGen provides the following types of logging:

- Basic logging for MashZone NextGen Server startup, shutdown and exceptions based on a configured logging level. See "Configure Logging for the MashZone NextGen Server" on page 1650 and "View the MashZone NextGen Server Log" on page 1741 for more information.
- Audit logging tracks the invocation of each mashable or mashup. See "Turn Audit Logging On or Off" on page 1652, "Purge the Audit Log" on page 1743 and "View the Audit Log for a Mashable, Mashup or App" on page 1742 for more information.

Configure Logging for the MashZone NextGen Server

The MashZone NextGen Server log, prestoserver.log, logs all exceptions from startup through shutdown. See "MashZone NextGen Logging" on page 1650 for links to additional types of logging you can use with MashZone NextGen.

Note: For clustered environments, updating logging configuration affects logging only for the MashZone NextGen Server where you are currently logged in. Generally, this is the behavior you want.

To change logging for all MashZone NextGen Servers in the cluster, update logging configuration for one server and copy the updated *MashZoneNG-install* /apache-tomee-jaxrs/conf/log4j.properties file to each of the other MashZone NextGen Servers in the cluster. You do not need to restart MashZone NextGen Servers.

To configure basic logging for the server

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Audits and Logs** section and click **Server Log**.
3. Edit any of the following properties:
   - *Root Log Warning* = the general logging level to use, such as `ERROR`. All exceptions for that level and above will be logged, so this contributes directly to how quickly logs may grow. `DEBUG` is the most verbose logging level.
   - **Log file path** = both the folder where the log files for the MashZone NextGen Server should be saved and the name to use for log files. This defaults to *tomcat-install* /logs/prestoserver.log.

     You can use an absolute path or a relative path. Relative paths are relative to the *web-apps-home* folder.
   - *Maximum log file size* = maximum size for a log file. Once a file has reached this size is it saved as a numbered backup, such as prestoserver.log.1 and a new log file is started.
   - *Data nucleus logging level* = This property should only be changed when requested by MashZone NextGen technical support to help debug specific issues. It is the logging level to use in the data mapping layer for MashZone NextGen.
   - *HTTP client logging level* = This property should only be changed when requested by MashZone NextGen technical support to help debug specific issues. It is the logging level to use for requests/responses between the MashZone NextGen Server and mashable information sources.
   - *NET SF logging level* = This property should only be changed when requested by MashZone NextGen technical support to help debug specific issues. It is the logging level to use for JSON serialization and deserialization.
   - *ACEGI security logging level* = This property should only be changed when requested by MashZone NextGen technical support to help debug specific issues. It is the logging level to use with the MashZone NextGen security layer.
   - *Apache logging level* = This property should only be changed when requested by MashZone NextGen technical support to help debug specific issues. It is the logging level to use with many of the third party libraries used in MashZone NextGen.
   - *Spring framework logging level* = This property should only be changed when requested by MashZone NextGen technical support to help debug specific issues. It is the logging level to use for server initialization, shutdown and the request/response pipeline for the MashZone NextGen Server.
4. If desired, click **Advanced Options** to set any of these properties:
   - *Log class for normal logging* = the java class that handles appending log entries to the log file. This defaults to org.apache.log4j.RollingFileAppender.

- *Layout class for normal logging* = the Java class that handles the layout pattern for entries to log files. This defaults to org.apache.log4j.PatternLayout.
- *Layout pattern for normal logging* = the expression defining the pattern for entries to the log file. This defaults to `%d %p [%c` - %m%n
- *Maximum normal log file backups* = how many log backups are kept. This defaults to seven.

The advanced properties are defined by Log4J, the underlying logging framework for MashZone NextGen. For more information, see http://logging.apache.org/log4j/1.2

5. Click **Save log settings**.

   This change becomes effective automatically within 60 seconds.

Turn Audit Logging On or Off

The Audit Log tracks the invocation of each mashable or mashup. This logging is disabled by default.

To turn audit logging on

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Audits and Logs** section and click **Audit Log**.

   A list of possible events for artifacts displays giving you fine grained control of what it logged:

   Some events apply to all artifacts (apps, mashables or mashups), such as the Created event. Others are specific to apps (event name begins with App) or to either mashables or mashups (event name begins with Service). Some user events are also listed.

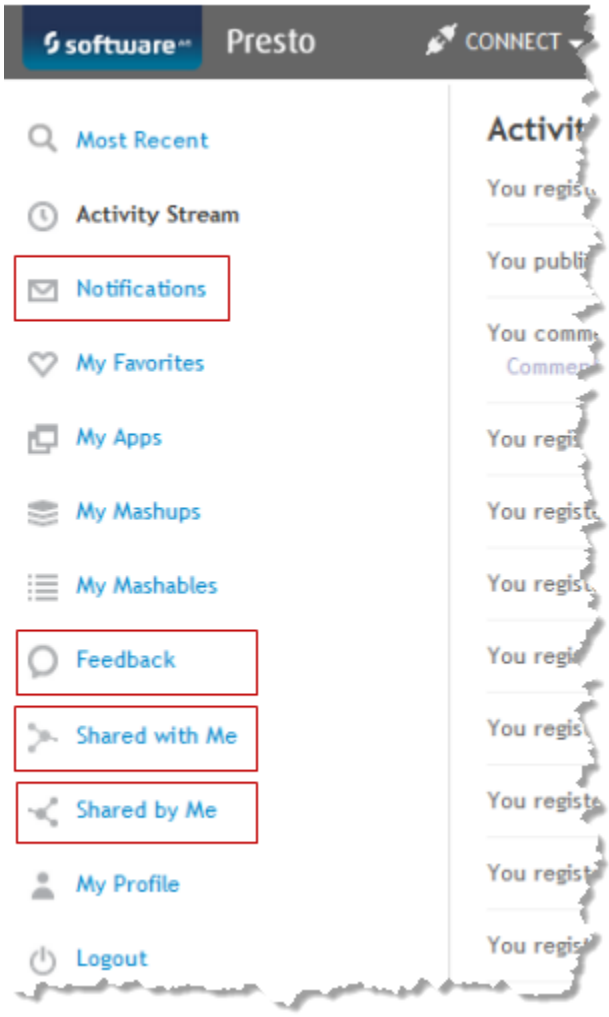
3. To turn audit logging on for a specific event, set the **Enable** option for that event.

> **Note:** Audit events are tracked in a table in the MashZone NextGen Repository, rather than being logged to a file. Because of the level of activity, this table can grow very large fairly quickly.
>
> To manage the size of this table, see "Purge the Audit Log" on page 1743.

4. To turn audit logging off for a specific event, clear this option for that event.

MashZone NextGen Notifications

MashZone NextGen sends notifications to users for many day-to-day events, such as comments that other users may have added to the apps, mashables or mashups that a user created, artifacts other users have shared with a user or notices when an app is accepted and is now published in the AppDepot. These notifications are visible in MashZone NextGen Hub from the home page:



You can also configure MashZone NextGen to send notifications as email messages.

To support email notifications

1. Configure an email server in MashZone NextGen. See "Configuring a Mail Server for MashZone NextGen" on page 1654 for instructions.
2. Optionally define the host and port to use for MashZone NextGen in links within email notices. See "Configure the MashZone NextGen Server with Custom Ports" on page 1638 for more information.
3. Turn on email notifications in the Admin Console:
   a. Click Admin Console in the MashZone NextGen Hub main menu.
   b. If needed, expand the **Server** section and click **Mail Notifications**.
   c. Set the **Turn ON Notifications** option and click **Save mail notification settings**.
4. If you are using your LDAP Directory as the MashZone NextGen User Repository, MashZone NextGen expects to find the email address for a user in the `mail` attribute in LDAP user entries. If your LDAP Directory stores email addresses in a different LDAP user attribute, you must configure MashZone NextGen to look for the correct LDAP user attribute. See "Update the User Email Attribute from LDAP" on page 1655 for instructions.

Configuring a Mail Server for MashZone NextGen

By default, MashZone NextGen sends notices for a variety of administrator and user actions within MashZone NextGen Hub. Users can access these notifications from the Home page. If you choose to send these notifications as email message, you must add connection configuration to the MashZone NextGen Server for your mail server.

1. Click Admin Consolein theMashZone NextGen Hubmain menu.
2. In the Server section, select**Mail Server**.
3. Complete the connection properties:
   - **Host** = your mail server domain name or host IP address.
   - **Port** = the port number for your mail server. This defaults to 25, for SMTP.
4. Set the **Requires authentication**if your SMTP server requires credentials and complete these properties:
   - **Username** = the user account the MashZone NextGen Server should use to connect to your mail server.
   - **Password** = the password the MashZone NextGen Server should use to connect to your mail server.
5. If your mail server uses a secure connection, choose the protocol for **Connection security**:

- `STARTTLS` = a transport layer security (TLS) that does not require a different port for the mail server.
- `SSL/TLS` = transport layer security (TLS) using the secure socket layer (SSL).

6. Click **Save mail server settings**.

Update the User Email Attribute from LDAP

When MashZone NextGen is configured to use your LDAP Directory as the User Repository, MashZone NextGen is configured by default to expect user email addresses in the `mail` attribute for LDAP user entries. If your LDAP Directory uses a different user attribute to store email addresses, you must update LDAP configuration information in MashZone NextGen to ensure that email notifications are successfully delivered.

To update the mail attribute

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **MashZone NextGen Repositories** section and click **User Repository - LDAP**.
3. Click **Advanced Options** and scroll down to find MashZone NextGen Query Properties.
4. Change the **User Email address** property to the name of the LDAP user attribute that contains user email addresses.
5. You may also want to update the list of attributes in **Export User Attributes as MashZone NextGen Attributes** and **Attributes Used in Wildcard Search** fields as these properties commonly include the `mail` attribute.
6. Click **Save the changes**.
7. Restart the MashZone NextGen Server to apply these updates.

Configure MashZone NextGen Connections to SharePoint

If you have the MashZone NextGen Add-On for SharePoint, you must configure connections in MashZone NextGen from the MashZone NextGen Server to SharePoint. You must also configure connections in SharePoint from SharePoint to the MashZone NextGen Server.

This configuration determines:

- How many MashZone NextGen Add-On for SharePoint licenses are needed.

  The MashZone NextGen Add-On for SharePoint license defines a maximum number of HTTP domains (domain-name/port combinations) for SharePoint. You can define any number of connections to the farm as a whole or to specific site collections within a given domain. See "Connection Patterns Between MashZone NextGen Servers and SharePoint" on page 1656 for more information and examples of how connections affect license usage.

- What mashups and apps are visible in SharePoint site collections to be added to Web Part pages in those sites.

  This is determined by the connections you define in SharePoint to MashZone NextGen Servers. Each connection must also have a valid license, but multiple connections can share a license based on the domains and ports used for your site collections and farms.

- What SharePoint site collections and Lists are visible in MashZone NextGen to be registered as mashable information sources or used as information sources in mashups.

  This is determined by the connections you define in MashZone NextGen from the MashZone NextGen Server to SharePoint farms and site collections. Users can also register SharePoint lists in SharePoint as mashables in MashZone NextGen, so this only affects users in MashZone NextGen.

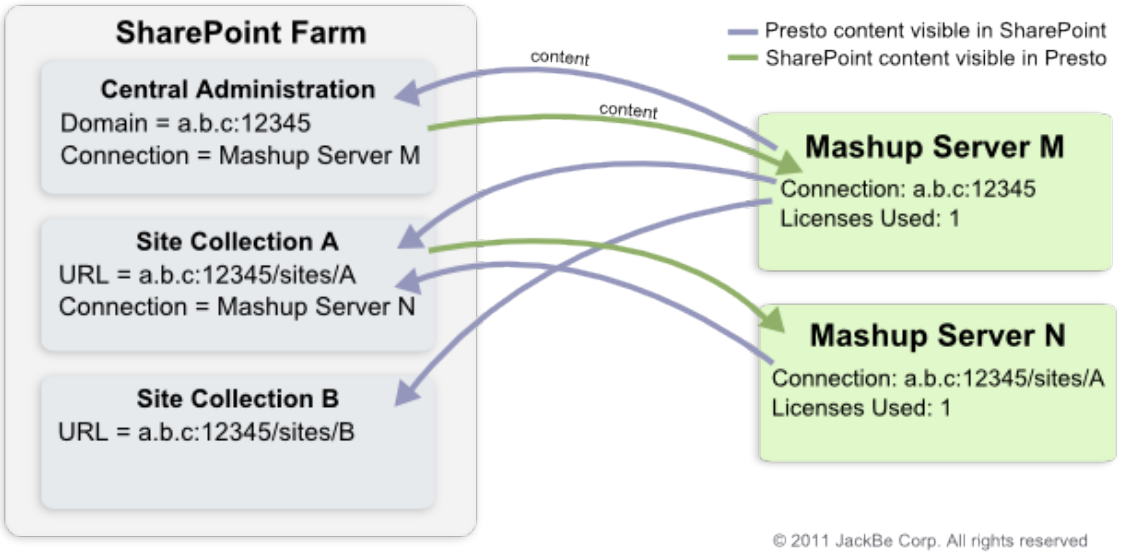
Connection Patterns Between MashZone NextGen Servers and SharePoint

How you define connections between MashZone NextGen and SharePoint determines what MashZone NextGen content is visible in SharePoint and what SharePoint content is visible in MashZone NextGen. This also determines how many MashZone NextGen Add-On for SharePoint licenses are required.

The simplest connection pattern is a single connection defined between one MashZone NextGen Server and an entire SharePoint farm. The connection is defined in SharePoint Central Administration to allow all site collections within the farm to share this connection.

This pattern works nicely when the domain and port number for all SharePoint site collections within a farm is the same as the domain and port number for SharePoint Central Administration.

In the following example, there is a single domain name and port for SharePoint Central Administration and all of the site collections within the SharePoint farm. A single connection to MashZone NextGen Server M is configured in the SharePoint Central Administration:



A corresponding connection to SharePoint Central Administration is defined in MashZone NextGen Server M. With this configuration, all site collections within the SharePoint farm can see and use mashups or apps hosted in MashZone NextGen Server M using a single MashZone NextGen Add-On for SharePoint license.

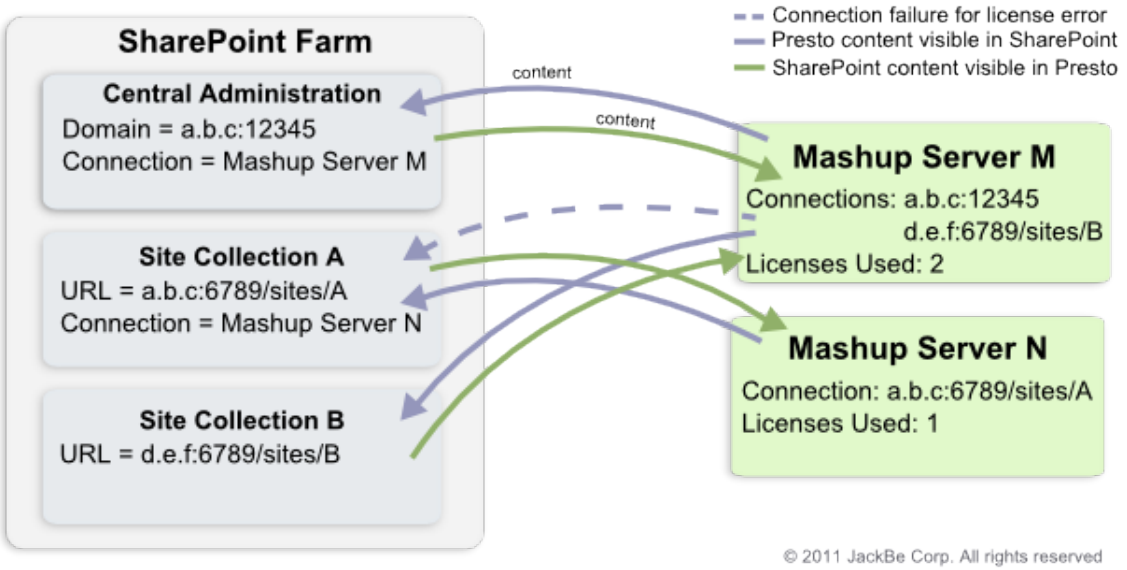
In MashZone NextGen, however, only the SharePoint Lists defined in Central Administration are visible. To make SharePoint Lists from Site Collection A and Site Collection B visible in MashZone NextGen, you would need to add separate connection configurations for each site collection in MashZone NextGen Server M. Even with multiple site collections, however, only a single MashZone NextGen Add-On for SharePoint license would be used because the domain and port for all these site collections is the same.

This example also defines a connection between a specific site collection and another MashZone NextGen Server. MashZone NextGen Server N has a connection defined to the SharePoint Site Collection A. A corresponding connection is defined in Site Collection A to MashZone NextGen Server N. This makes the mashup and apps in MashZone NextGen Server N visible in any SharePoint site within the Site Collection A and makes any SharePoint lists in any site in Site Collection A visible to MashZone NextGen users logged into MashZone NextGen Server N.

More complex connection configurations are needed when Central Administration and site collections within a SharePoint farm use different domains or port numbers.

The following example again has one SharePoint farm with Central Adminisration plus two site collections. Site Collection A uses the same domain as Central Adminisration, but a different port and Site Collection B uses a different domain and port number.

SharePoint Central Adminisration again has a single connection defined to MashZone NextGen Server M and both Site Collection A and B use this connection configuration. Because of the different domain names or port numbers, however, several connections must be configured in MashZone NextGen Server M:



- The connection to the SharePoint Central Adminisration in MashZone NextGen Server M is required to allow mashups and apps in MashZone NextGen Server M to be visible in the SharePoint farm. This works in conjunction with the connection configuration in the SharePoint Central Adminisration to MashZone NextGen ServerM.

  This connection uses one license.

- A second connection to Site Collection B is required in MashZone NextGen Server M to allow mashups and apps from MashZone NextGen Server M to be visible in Site Collection B. Because Site Collection B inherits the shared farm-wide connection configuration, no connection configuration is required in this SharePoint site.

  This connection uses a second license.

- In this example, Site Collection A inherits the shared, farm-wide connection to MashZone NextGen Server M from Central Adminisration configuration. But no mashups or apps from MashZone NextGen Server M will be visible in Site Collection A because of license errors. Site Collection A cannot use the license defined for Central Adminisration because the port number is different.

  To make this connection work properly, another connection must be added to MashZone NextGen Server M for Site Collection A. This would use a third license.

If content in SharePoint Central Adminisration is not important, this could also simply be configured as connections at each site collection to the relevant MashZone NextGen Servers and connections in each MashZone NextGen Server to the relevant SharePoint site collections. For example:

SharePoint Site Collections	MashZone NextGen Servers
Site Collection A has connection configuration to: ■ MashZone NextGen Server M ■ MashZone NextGen Server N	MashZone NextGen Server M has connections configuration to: ■ Site Collection A ■ Site Collection B
Site Collection B has connection configuration to MashZone NextGen Server M.	MashZone NextGen Server N has connection configuration to Site Collection A.

Add SharePoint Connections to the MashZone NextGen Server

You must add connection information for SharePoint to the MashZone NextGen Server*before* you can add connection information for the MashZone NextGen Server to SharePoint. To determine what connections you should configure, see "Connection Patterns Between MashZone NextGen Servers and SharePoint" on page 1656 for examples and more information.

Each SharePoint connection in the MashZone NextGen Server defines the URL and credentials to a specific SharePoint site collection and assigns a connection name. The endpoint for the connection may also be SharePoint Central Adminisration for an entire SharePoint farm.

The credentials you enter for a connection to a SharePoint site collection are known as the *service account*. The service account creates global MashZone NextGen attributes. Users can also define their own credentials for a SharePoint connections, which create MashZone NextGen attributes for that user.

You configure which set of credentials can be used in mashables or mashups that connect to each SharePoint site collection. This determines what options users have when they register mashables or use SharePoint blocks in mashups. Connections can:

- Always use the *service account* credentials.
- Always use *user* credentials.
- Default to the *service account* credentials, but users can *override* this and use their own credentials or use the current user's credentials.

To add a SharePoint connection

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Add-Ons section and click **SharePoint**.
3. Click **Add a new SharePoint instance**.
4. Complete the following properties:

- **Name** = a name to identify this SharePoint connection.

  Note: Connection names *must* be unique in the MashZone NextGen Repository.

- **Endpoint** = the URL to connect to the root site in SharePoint for the site collection or Central Administration for a SharePoint farm. If single sign-on is configured for MashZone NextGen Add-On for SharePoint, this is also the SharePoint server that the MashZone NextGen Server works with to validate SSO tokens when users in SharePoint connect to MashZone NextGen.

  Note: If this URL uses HTTPS for secure connections, this uses digital certificates in the *default JVM truststore* which is typically in the *java-install* /jre/lib/security/cacerts folder. This may *not* be the same truststore that has been configured for the MashZone NextGen Server.

- **Authentication Type** = the authentication protocol between the MashZone NextGen Server and SharePoint when mashables and mashups are run. Choose one of:
  - `NTLM`: the default authentication protocol. This uses NTLM version 2 with the Windows domain plus basic credentials.
  - `Basic`: basic username and password credentials for authentication.
  - `SSO`: choose this if SharePoint uses an agent-based single sign-on solution, such as SiteMinder, rather than the built-in, token-based SSO solution for MashZone NextGen Add-On for SharePoint.
- **Service Account Username** = the user name for the global credentials that the MashZone NextGen Server should use to connect to this SharePoint site collection. This is required for NTLM or Basic authentication only.

  Note: The service account credentials may also be the default credentials or the only credentials for this connection, based on the **Default** and **Override** options.

- **Service Account Password** = the password for the global credentials that the MashZone NextGen Server should use to connect to this SharePoint site collection. This is required for NTLM or Basic authentication only.

  Note: The service account credentials may also be the default credentials or the only credentials for this connection, based on the **Default** and **Override** options.

- **Domain** = this is only required when the authentication type is `NTLM`. This is the Windows domain for the service account credentials.
- **SSO Token Server Name** = the name that will be passed in requests from SharePoint to the MashZone NextGen Server along with SSO tokens. This name identifies the SharePoint server that should validate SSO tokens for the MashZone NextGen Server.

> **Note:** This field is used *only* for the built-in, token-based single sign-on solution for MashZone NextGen Add-On for SharePoint. It applies only to requests from SharePoint to the MashZone NextGen Server and does not affect authentication for mashables or mashups that use SharePoint as an information source.

This name *must* match the SSO token server name assigned in SharePoint Central Administration for the connection that SharePoint uses to connect to the MashZone NextGen Server. See the "Configure Authentication for P4S 2010" on page 1379 or "Configure Authentication for P4S 2007" on page 1400 for more information.

> **Note:** Enabling the MOSS Single Sign-On feature in SharePoint 2007 may affect security for SharePoint. Contact your SharePoint administrator or see Microsoft documentation for more information.

5. If users *must always* have their own SharePoint credentials to connect to this site collection, clear the **Default to service account credentials** option.

   This option is set by default. If the **Always use service account credentials** option is *not* set, users can create their own SharePoint credentials in MashZone NextGen when they register mashables SharePoint Lists or use SharePoint blocks in Wires and these credentials override the service account.

6. Set the **Always use service account credentials** option if the service account *must always* be used to connect to this SharePoint site collection.

   This option is clear by default. This allows users to create their own SharePoint credentials in MashZone NextGen when they register mashables SharePoint Lists or use SharePoint blocks in Wires. If the **Default** option is also clear, then users are required to supply their SharePoint credentials.

7. Click **Save**.

   If you have provided credentials, MashZone NextGen validates this connection and adds it to the list of available connections along with an icon indicating the status:

   - = the connection is valid
   - = an error occurred verifying this connection

You can also **Edit** or **Delete** individual SharePoint connections from this list.

BigMemory for Caching, Connections and MashZone NextGen Analytics

By default MashZone NextGen uses local memory for caching and the MashZone NextGen Analytics In-Memory Stores that MashZone NextGen Analytics creates. This uses BigMemory as a local client that is installed with MashZone NextGen and requires only your MashZone NextGen license. In specific cases, you must also install BigMemory Servers on one or more additional hosts and configure MashZone NextGen

and the Integrated MashZone Server to work with them. MashZone NextGen requires BigMemory Servers with a BigMemory license to support:

- Significant, extensible amounts of memory, most commonly for very large datasets used with MashZone NextGen Analytics.

  BigMemory Servers can be deployed in clusters, also known as Terracotta Server Arrays, that can easily be extended for scalable memory requirements.

- Access to *off-heap* memory.

  BigMemory Servers also can manage memory outside of heap both for better scalability and performance improvements.

- Access to In-Memory Stores created and populated dynamically by external systems.

  BigMemory manages the In-Memory Stores created dynamically by other systems and makes connection information available to MashZone NextGen through the Terracotta Management Console (TMC) to allow MashZone NextGen to work with this data. Apama, for example, dynamically creates distributed stores for the Apama MemoryStore which MashZone NextGen can connect to and query.

- Distributed caching when MashZone NextGen is deployed in clusters.

  With clusters, some of the interal MashZone NextGen caches must be distributed and managed by BigMemory Servers.

- MashZone feeds that use BigMemory connections as a data source.

See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for instructions on configuring MashZone NextGen to work with BigMemory Servers.

See "Working with MashZone NextGen Analytics In-Memory Stores" on page 857 for an overview and links to additional information on MashZone NextGen Analytics In-Memory Stores.

You also need to provide configuration and connection information for In-Memory Stores that are created by other systems or stores created by MashZone NextGen Analytics that need different settings than the defaults. For more information on these tasks, see:

- "Declare BigMemory Stores for MashZone NextGen Analytics" on page 1667
- "Manage Dynamic BigMemory Stores for MashZone NextGen Analytics" on page 1671.

See "Caching for the MashZone NextGen Server" on page 1662 for an overview of MashZone NextGen caching. For caching configuration, see "Configuring Mashable/Mashup Response Caching" on page 1673 and "Distributed Caching for MashZone NextGen Clusters" on page 1663.

Caching for the MashZone NextGen Server

The MashZone NextGen Server caches artifact metadata, for internal purposes, and optionally caches mashable and mashup responses to improve performance. By default,

caches are stored in local memory. See "Artifact Caching" on page 1663 and "Response Caching" on page 1663 for details.

The Integrated MashZone Server also uses local memory for internal caches. See "Tune Memory/Caching for the Integrated MashZone Server" on page 1697 for information on caching for the Integrated MashZone Server.

All MashZone NextGen caches plus the MashZone NextGen Analytics In-Memory Stores, can be distributed when MashZone NextGen is deployed in clusters. See "Distributed Caching for MashZone NextGen Clusters" on page 1663 for an overview of distributed caching for MashZone NextGen.

Caches for the the Integrated MashZone Server do not support distributed caching and must remain in local memory.

Artifact Caching

Artifact caching caches metadata for mashables, mashups and apps when they are run in MashZone NextGen Hub, the AppDepot, the MashZone NextGen Mobile apps or in any external destination such as SharePoint, portals or web pages. For example, the first time an app is viewed, the specification for that app is retrieved from the MashZone NextGen Repository and cached. Subsequent requests use the cached specification.

Because updates to artifacts are typically infrequent, this cache is long-lived. It is not persisted to disk. Cache entries are flushed only when an artifact is updated or when the server hosting the cache is restarted. No additional configuration is required to enable local artifact caching for MashZone NextGen.

Response Caching

Response caching caches the responses from mashables and mashups when they are run. This is a short lived cache that caches response data based on the unique signature of each request to mashables or mashups.

Configuration for response caching gives you fine grained control for which mashables and mashups use response caching and the expiration periods for their cache entries. See "Configuring Mashable/Mashup Response Caching" on page 1673 for instructions.

Distributed Caching for MashZone NextGen Clusters

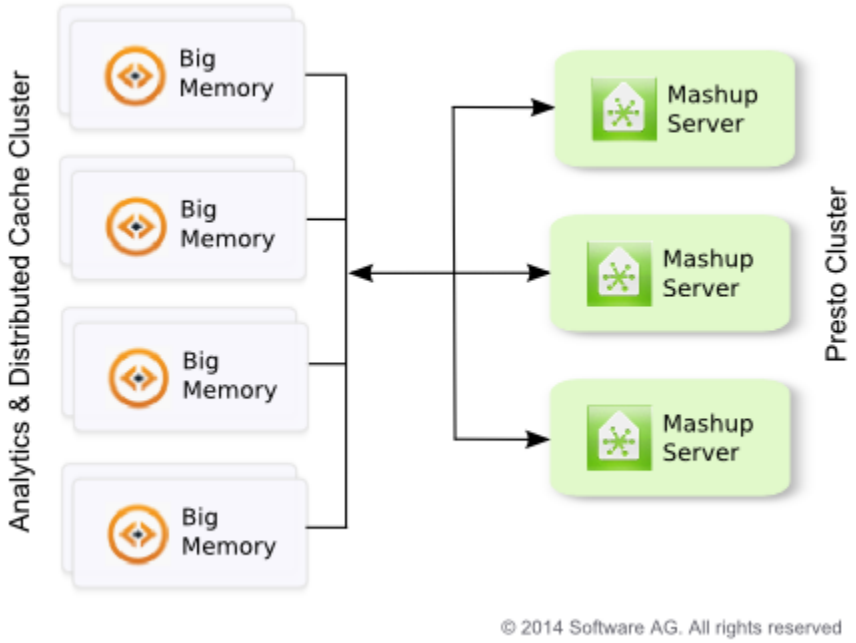
When MashZone NextGen is deployed in clusters, both artifact caching and the MashZone NextGen Analytics In-Memory Stores*must* be distributed to maintain cache integrity. Response caching, however, can be left in local memory for each MashZone NextGen Server or it can be distributed.

In many environments, local caching provides both good performance and acceptable cache integrity for response caching. Local caching is "eventually consistent", but can result in visible differences as cached responses are not guaranteed to be identical for different cluster members. For environments that cannot tolerate any loss of cache integrity, distributed response caching is recommended.

Note: Distributed caching is only available if you purchase and deploy BigMemory Servers.

Caches for the the Integrated MashZone Server do not support distributed caching and must remain in local memory.

You use BigMemory Servers to handle distributed caching for MashZone NextGen. When this is combined with MashZone NextGen clusters, the high-level architecture looks like this:



With BigMemory Servers, data for both the MashZone NextGen Analytics In-Memory Stores and most MashZone NextGen caches can use the total off-heap memory configured for the cluster plus any heap and off-heap memory configured for the MashZone NextGen local host.

The BigMemory Servers manage consistency and memory across the cluster. They also support failover, with mirror servers, for high availability and many other advanced capabilities that may be useful for enterprise production systems.

To configure distributed caching, see "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for set up instructions.

Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores

You can configure MashZone NextGen to work with one or an array of BigMemory Servers to provide additional memory, provide reliability and support specific other features. See "BigMemory for Caching, Connections and MashZone NextGen Analytics" on page 1661 for more information on features that require BigMemory servers.

To configure BigMemory Servers for MashZone NextGen

1. Copy your license for BigMemory to MashZone NextGen and update MashZone NextGen startup scripts:

   a. Copy the license file terracotta.key to the *MashZoneNG-install* /apache-tomee-jaxrs/conf folder.

   > **Note:** If MashZone NextGen is deployed in a cluster, you must copy this file to every cluster member.

   b. Add this Java system property to the MashZone NextGen startup script :

   ```
   -Dcom.tc.productkey.path=MashZoneNG-install/apache-tomee-jaxrs/conf/terracotta.key
   ```

   Add this property to either:

   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.bat file for Windows systems, or
   - *MashZoneNG-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX systems.

   > **Note:** If MashZone NextGen is deployed in a cluster, you must update the startup scripts for every cluster member.

2. Edit the ehcache.xml file for MashZone NextGen in a text editor of your choice. The location depends on your deployed environment:

Only One MashZone NextGen Server	*MashZoneNG-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/ehcache.xml
A MashZone NextGen Cluster and Shared External Configuration Folder	*MashZoneNG-config* /ehcache.xml For more information in a clustered environment, see "Setting Up an External MashZone NextGen Configuration Folder" on page 1777.
A MashZone NextGen Cluster With No Shared Configuration	You must update *MashZoneNG-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/ehcache.xml for each cluster member.

3. Find the line in ehcache.xml with `<terracottaConfig>` that is commented out like this:

   ```
   <!-- <terracottaConfig url="localhost:9510" /> -->
   ```

   Remove the comment markers and change the `url` attribute to the host (or IP address) and port for the BigMemory server(s) you installed. For example:

```
<terracottaConfig url="tcHost1:9510" />
```

Note: There are several ways to identify one or more BigMemory servers for MashZone NextGen. See BigMemory documentation for more information.

4. Find the line in ehcache.xml with `<terracotta>` that is commented out and uncomment this line for each of the following named `<cache>` elements:
   - `RAQL_DATA_CACHE` = the MashZone NextGen Analytics In-Memory Stores
   - `SEARCH_RESULTS_CACHE` = one part of the MashZone NextGen Artifact cache.
   - `SERVICES_BY_ID_CACHE` = = one part of the MashZone NextGen Artifact cache.
   - `SERVICE_RESPONSE_CACHE` = the MashZone NextGen Response cache for mashables and mashups. This is optional. Update this cache *only* if you want it to be distributed.

   This `<terracotta>` element allows the In-Memory Store and MashZone NextGen caches to use heap and off-heap memory in both the local host and BigMemory hosts. This combined memory is managed by BigMemory.

   For more information on the `<terracotta>` element, see *Distributed Configuration* topics in BigMemory documentation.
5. Save these changes to ehcache.xml.

   Important: For clusters where this configuration file is not stored in a shared external folder, copy this file to the same location for each MashZone NextGen cluster member.
6. Optionally, update configuration for dynamic In-Memory Stores that MashZone NextGen Analytics creates to work with BigMemory servers:
   a. Edit the dynamiccache.xml file in these locations (based on your deployed environment):

Only One MashZone NextGen Server	*MashZoneNG-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/dynamiccache.xml
A MashZone NextGen Cluster and Shared External Configuration Folder	*MashZoneNG-config* /dynamiccache.xml For more information in a clustered environment, see "Setting Up an External MashZone NextGen Configuration Folder" on page 1777.
A MashZone NextGen Cluster With No Shared Configuration	You must update *MashZoneNG-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/dynamiccache.xml for each cluster member.

b. Add a `<terracotta>` element inside the `<cache>` element.

c. Provide URLs or other connection information as needed.

For more information on configuration for the `<terracotta>` element, see BigMemory documentation.

d. Save your changes.

7. Optionally, update configuration for any declared In-Memory Stores to work with these BigMemory server(s):

a. In the configuration files for your declared In-Memory Stores, add a `<terracotta>` element inside the corresponding `<cache>` element.

b. Provide URLs or other connection information as needed.

For more information on the `<terracotta>` element, see BigMemory documentation.

c. Save changes to these files.

d. Upload these configuration changes. See "Modify a Declared In-Memory Store" on page 1670 for instructions.

8. Start BigMemory Server(s).

9. If needed, adjust memory configuration for the local MashZone NextGen host. See "Memory Configuration for the MashZone NextGen Server" on page 1627 for instructions.

10. Restart MashZone NextGen. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Declare BigMemory Stores for MashZone NextGen Analytics

Declaring In-Memory Stores for MashZone NextGen Analytics allows MashZone NextGen to connect to stores in BigMemory that may have been created and populated by other systems. Declared stores also allow you to fully control configuration for In-Memory Stores created and populated by MashZone NextGen Analytics.

Note: You can also define connections to in-memory stores that are created dynamically by external systems. See "Manage Dynamic BigMemory Stores for MashZone NextGen Analytics" on page 1671 for more information.

See "Declare a New In-Memory Store" on page 1667, "Modify a Declared In-Memory Store" on page 1670 and "View Details for Declared In-Memory Stores" on page 1670 for more information.

Declare a New In-Memory Store

1. Define configuration for one or more In-Memory Stores in a cache configuration file for BigMemory (an ehcache.xml file).

Tip: It is a best practice to change the default file name ehcache.xml for this configuration file to something more meaningful, such as myCRM-cache.xml. This makes it easier to identify when multiple configuration files are uploaded to MashZone NextGen.

a. Add a `name` attribute to the `<ehcache>` element and assign a unique name.

   This is the *cache manager name* which *must* be unique for this MashZone NextGen Server. It should consist solely of letters, numbers, underscores(_) or dashes (-).

b. Add a `<cache>` element for each store you need to declare. The following example shows some common properties. See BigMemory documentation for more information.

   Note: You can find this example configuration file, sample-cache.xml, for declared stores in the *MashZoneNG-install* /prestocli/raql-samples folder.

```
<ehcache name="sample-cache" >
 <diskStore path="java.io.tmpdir"/>
...
 <cache name="StocksDeclCache"
        maxBytesLocalHeap="50M"
        memoryStoreEvictionPolicy="LRU"
        timeToIdleSeconds="0"
        timeToLiveSeconds="0">
 </cache>
...
</ehcache>
```

c. If this In-Memory Store will be created and populated by MashZone NextGen, add a `<searchable allowDynamicIndexing="true">` element to `<cache>` to support dynamic searching.

   For example:

```
<ehcache name="sample-cache" >
 <diskStore path="java.io.tmpdir"/>
...
 <cache name="StocksDeclCache"
        maxBytesLocalHeap="50M"
        memoryStoreEvictionPolicy="LRU"
        timeToIdleSeconds="0"
        timeToLiveSeconds="0">
  <searchable allowDynamicIndexing="true"/>
 </cache>
...
</ehcache>
```

d. If this In-Memory Store will be populated by external systems with datasets that are Java objects, add `<searchable>` to the `<cache>` element and define a `<searchAttribute>` with the name, datatype and extractor class for each property in these Java objects that will contain data.

   For the `class` attribute, use the `net.sf.ehcache.search.AttributeExtractor` interface provided in the BigMemory Search API or an implementation class of `AttributeExtractor`. See BigMemory documentation for details.

Important: MashZone NextGen is only able to access searchable attributes of datasets stored by external systems. For Apama used as external system, search attributes are no more required. If search attributes are not explicitly specified and the dataset is stored using RAQL, all attributes are made searchable automatically.

Since version 9.9 MashZone NextGen supports the native Apama RowValue format. MashZone NextGen can consume RowValues stored by Apama and convert them into the RAQL record format. In case of caches written by Apama searchable attributes are no more needed for accessing the data at all but they are still required for processing filters, aggregations and sorting directly in BigMemory.

See "Dataset Paths, Names and Datatypes" on page 721 and "Valid RAQL Datatypes" on page 772 for more information on names and datatypes.

e. Save this file.

2. Copy the JAR file containing the classes used as search attributes to extract data from the dataset in this cache to *MashZoneNG-config* /lib.

   See documentation for the external system that created this dynamic store to determine what JAR files are needed. For Apama, see documentation on the MemoryStore.

   The default location for this folder in MashZone NextGen is *MashZoneNG-install* / apache-tomee-jaxrs/pretso/WEB-INF/lib. If MashZone NextGen is deployed in a cluster, however, this location may be a separate external folder. For more information, see "Setting Up an External MashZone NextGen Configuration Folder" on page 1777.

3. Restart the MashZone NextGen Server. For instructions, see "Start and Stop the MashZone NextGen Server" on page 1570.
4. Click Admin Console in the MashZone NextGen Hub main menu.
5. Click **Server** to expand this section of the Administration menu.
6. Click **BigMemory**.
7. Open the **BigMemory Cache** tap.

   The Big Memory Cache tab lists any In-Memory Store configuration files that have already been upload.

8. Click **Register a new EhCache Configuration File**.
9. Enter the name assigned to `<ehcache>` in this configuration file (in step 1) as the **Big Memory Data Source Name**. This name is used as a prefix for all stores defined in this configuration file to uniquely identify each store.

   Important: Data source names *must* be unique for this MashZone NextGen Server. They should contain only letters, numbers, underscores (_) or dashes (-).

If any of the declared In-Memory Stores for this connection have data populated by external systems (not through `<storeto>`), the data source name *must also* match the name assigned to the `<ehcache>` element in the configuration file.

10. Click **Browse** to find and select the *Cache Configuration File*ehcache.xml you created in step 1.
11. Click **Add this file**.

    The file is uploaded to the MashZone NextGen Repository in the standard path bigmemory/caches/*file-name* and shown in the list by data source name. The URL shown is the relative path in MashZone NextGen to this resource.

Note: Administrators can also manage resources files in the Admin Console. See "Manage Files for MashZone NextGen Features or Artifacts" on page 1745 for more information.

Modify a Declared In-Memory Store

1. Update the configuration file for a declared In-Memory Store as needed.

   For example, you may need to add configuration to allow an In-Memory Store to use memory in external BigMemory hosts when you add servers or deploy MashZone NextGen in staging or production environments.
2. Click Admin Console in the MashZone NextGen Hub main menu.
3. Click **Server** to expand this section of the Administration menu.
4. Click **BigMemory**.
5. Open the **BigMemory Cache** tap.

   The **Big Memory Cache** tab lists any In-Memory Store configuration files that have already been upload.
6. Select the existing BigMemory data source for this store and click **Delete**.
7. Add this data source with the updated configuration file. See "Declare a New In-Memory Store" on page 1667 for instructions.

View Details for Declared In-Memory Stores

To see configuration information for declared In-Memory Stores:

- Click Admin Console in the MashZone NextGen Hub main menu.
- Click **Server** to expand this section of the Administration menu.
- Click **Big Memory**.
- Open the **BigMemory Cache** tap.

This lists any configuration files for declared In-Memory Store that have already been upload.

- Click the **Title** for a configuration file to see detailed information for the In-Memory Stores declared in that file.
- To see the configuration file contents, click the **URL** for that file.

Manage Dynamic BigMemory Stores for MashZone NextGen Analytics

You must define connections and identify configuration information for BigMemory stores that are created by and store data from external systems and then are used as In-Memory Stores in MashZone NextGen Analytics. For in-memory stores that are created dynamically by other systems, MashZone NextGen retrieves configuration and connection information from the Terracotta Management Console (TMC) that manages the host BigMemory Server.

Note: You can also define connections to external in-memory stores that are not created dynamically. See "Declare BigMemory Stores for MashZone NextGen Analytics" on page 1667 for more information.

For information on the requirements for in-memory stores that act as dynamic external stores for MashZone NextGen Analytics. See "External versus Internal Datasets" on page 859. For instructions on adding and managing external dynamic store configuration, see "Add an External Dynamic In-Memory Store Connection" on page 1671 and "Delete External Dynamic In-Memory Store Connections" on page 1672.

Add an External Dynamic In-Memory Store Connection

To add an external dymamic in-memory store connection

1. Verify that the Terracotta Managment Console (TMC) that manages the BigMemory Server hosting this dynamic store is running and that the store exists.

   You should also verify that the dynamic store meets minimum requirements for MashZone NextGen. See "External versus Internal Datasets" on page 859 for details.

2. Copy the JAR file containing the classes used as search attributes to extract data from the dataset in this store to *MashZoneNG-install* /lib.

   See documentation for the external system that created this dynamic store to determine what JAR files are needed. For Apama, see documentation on the MemoryStore.

   The default location for the destination folder in MashZone NextGen is *MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/lib. If MashZone NextGen is deployed in a cluster, however, this location may be a separate external folder. For more information, see "Setting Up an External MashZone NextGen Configuration Folder" on page 1777.

3. Restart the MashZone NextGen Server. For instructions, see "Start and Stop the MashZone NextGen Server" on page 1570.

4. Click Admin Console in the MashZone NextGen Hub main menu.
5. Click **Server** to expand this section of the Administration menu.
6. Click **Big Memory**.
7. Open the **Terracotta Management Server** tab.

   This tab lists connections to any existing external dynamic In-Memory Stores.
8. Click **Register a new Terracotta Management Server**.
9. Enter a unique **Big Memory Data Source Name** for this connection to the dynamic external cache that will act as an In-Memory Store.
10. Enter the domain and port, or IP address and port for the Terracotta Managment Server. For example: `localhost:9889`.
11. Enter the **Terracotta Management Server Connection Name**.

    Connection names cannot include periods (.), spaces or other common symbols or punctuation characters.
12. Enter the name of the **Cache Manager** for this cache. This name is assigned by the external system that created the cache in BigMemory.

    Tip: Cache Manager names are a best practice for dynamic external stores. If the external system does not assigne a cache manager name, BigMemory uses a default name which can lead to name collisions and errors.
13. If the TMC requires SSL for connections, change the **Security type** to `SSL`.
14. You can enter an **Username** and a **Password** optionally.
15. Click **Add this external cache** to save this connection.

Delete External Dynamic In-Memory Store Connections

To delete the configuration for an external dynamic In-Memory Store

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Server** to expand this section of the Administration menu.
3. Click **Big Memory**.
4. Open the **Terracotta Management Server** tab.

   This tab lists connections to any existing *Terracotta Management Server*.
5. Click **Delete** for the specific connection you want to delete.

Configuring Mashable/Mashup Response Caching

In non-clustered environments, each MashZone NextGen Server has a local Response Cache that is disabled by default. You must have MashZone NextGen administrator permissions to enable response caching.

When MashZone NextGen is deployed in clusters, you can continue to use local response caching or you can use distributed caching for the cluster. Distributed caching requires a cluster or server array of Terracotta BigMemory. You can install this cluster and then configure configure distributed caching for responses. See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for instructions.

Enable and Configure Response Caching

For response caching to work, you must complete these steps:

1. Enable response caching for the MashZone NextGen Server.

   Response caching configuration for individual mashables or mashups is ignored until a MashZone NextGen administrator enables the Response Cache for the MashZone NextGen Server.

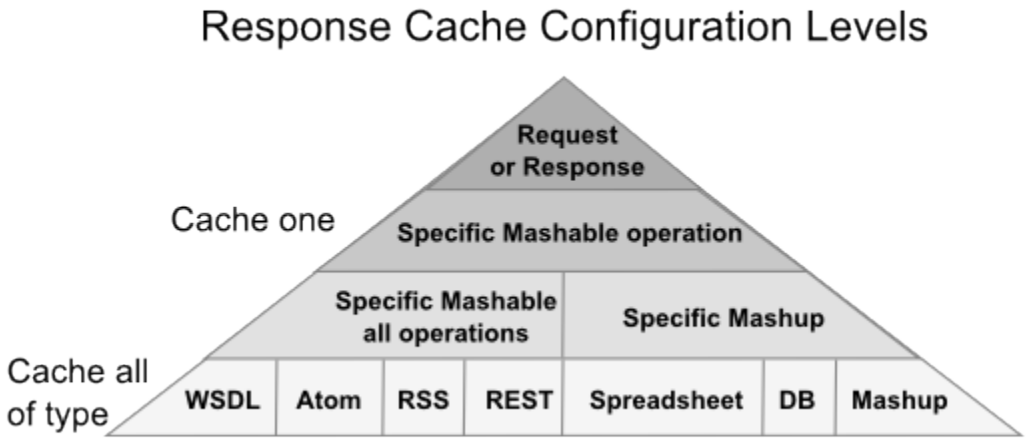
   a. Click Admin Console in the MashZone NextGen Hub main menu.

   b. Expand the **Platform Features** section and click **Caching**.

   c. Set the **Enable Caching** option to turn response caching on.

   d. Click **Save**.

2. Configure caching requirements at one or more of the following levels:



   Each level overrides caching configuration at any lower level both to determine whether caching occurs and to set the expiration period for a specific cached response. You can also set a default expiration period. See the "Response Caching Example" on page 1677 for an illustration of the effects of caching configuration.

If you want to cache responses in most cases, it is best to enable caching and configure the expiration for entire types of mashables or mashups.Then disable or change expirations for the exceptions individually. See "Cache Responses by Default and Disable Exceptions" on page 1674 for instructions.

If you only need to cache responses for specific mashables or mashups, it is best to leave caching disabled for all mashable types and mashups and enable caching only for specific exceptions. See "Cache Responses for Exceptions Only" on page 1675 for instructions.

You can also override cache settings for individual requests or responses. See "Controlling Response Cache Entries Dynamically" on page 1676 for more information.

Cache Responses by Default and Disable Exceptions

This pattern enables response caching for all mashables of one or more types or for all mashups and disables caching only for specific mashables or mashups. You can also use this pattern to enable response caching for individual mashables and disable caching for specific operations.

Note: Only MashZone NextGen administrators can configure response caching for all mashables of one type or all mashups. Only owners or MashZone NextGen administrators can configure response caching for a specific mashable or mashup.

1. Enable response caching for all mashables of one type or for all mashups:
   a. Click Admin Console in the MashZone NextGen Hub main menu.
   b. Expand the **Platform Features** section and click **Caching**.
   c. If desired, change the **Default Maximum Age** for response cache entries. This defaults to 5 minutes. Enter a number and/or change the time interval.
   d. Set the caching option for a mashable type or for mashups to enable caching for all mashables of that type or for all mashups.

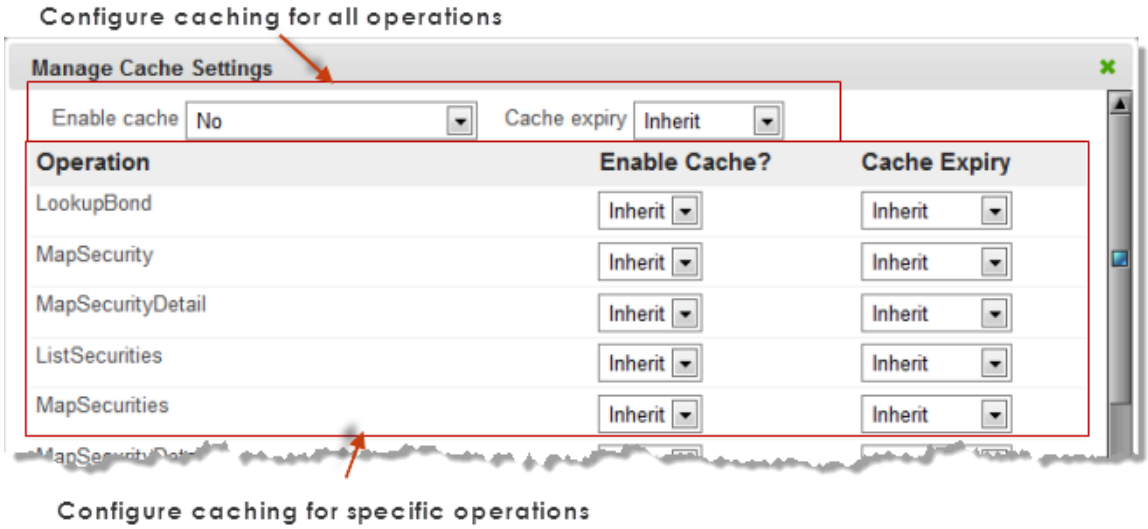
      For example, to enable caching for all mashups, set the **Enable caching for Mashups** option.
   e. If desired, set the maximum age for response cache entries for all mashables of this type or for all mashups. Enter a number or change the time interval.

      This default overrides the global default expiration period. Mashables or mashups can inheirit this expiration period or override it.
   f. Enable caching for mashups or other mashable types as desired.
   g. Click **Save**.
2. For mashables or mashups that should *not* have responses cached or that should use a different expiration period:

a. Use search or browse to find the mashable or mashup you want and open its artifact page.

b. Click **Manage** > **Cache**.

You can control response caching for all operations for a mashable, for mashups or for specific operations of a mashable as shown in the following example:



c. To turn response caching off for all operations for this mashable or mashup, set **Enable cache** to `No`.

d. To leave response caching on but change the expiration period for cache entries for all operations of this mashable or mashup, change the **Cache expiry** from `Inherit` to a specific value.

e. If needed, select specific operations and turn response caching on or update their expiration periods in the corresponding operation fields.

f. Click **Close**.

You can also update cache settings to multiple artifacts using the bulk update feature. See "Update Caching for Multiple Mashables or Mashups" on page 100 for instructions.

Cache Responses for Exceptions Only

With this pattern, you leave response caching disabled for mashable types and all mashups. Then enable response caching only for those artifacts and operations where it is needed.

Note: You must own the mashable or mashup or be a MashZone NextGen administrator to update response caching configuration for a mashable or mashup.

To enable caching for an individual mashable or mashup:

1. Use search or browse to find the mashable or mashup you want and click it to open its artifact page.
2. Click **Manage >** **Cache**.
3. Change **Enable cache** from `Inherit` to `Yes` for either:
   - All operations for the mashable or mashup.
   - A specific mashable operation.
4. If needed, set the expiration period for cache entries for this operation. Change the **Cache expiry** from `Inherit` to a specific value.
5. Update any other operations that should be cached.
6. Click **Close**.

Or use the bulk update feature to enable caching for several artifacts at once. See "Update Caching for Multiple Mashables or Mashups" on page 100 for instructions.

Controlling Response Cache Entries Dynamically

You can use HTTP headers in requests or responses to provide individual control of cache entries that override all other cache configuration. This uses the HTTP `Cache-Control` header.

You can add HTTP headers to requests to run mashables or mashups using MashZone NextGen Connect or the MashZone NextGen REST API. To set caching headers in responses, wrap requests to run mashables in a mashup and use EMML statements in the mashup to add the HTTP headers to the response. See "Adding HTTP Headers to the Mashup Result" on page 607 for instructions.

Where you add this header and the specific value determines the effect:

- To ensure that the response is no older than a specific number of milliseconds, set one of the following HTTP headers in a *request* to invoke a mashable or mashup:
  - `CACHE-CONTROL: "max-age=`*`number-of-seconds`*`"`
  - `max-age=`*`number-of-seconds`*
- To set the maximum age of a new cache entry created for a specific response, set one of these HTTP headers in the mashable or mashup *response*:
  - `CACHE-CONTROL: "max-age=`*`number-of-seconds`*`"`
  - `max-age=`*`number-of-seconds`*
- To force the MashZone NextGen Server to discard the current cache entry and invoke a mashable or mashup, set one of these HTTP headers in the mashable or mashup *request*:
  - `CACHE-CONTROL: no-cache`

- `no-cache`

- To ensure the current response is *not* cached, set one of these HTTP headers in the mashable or mashup *response*:
  - `CACHE-CONTROL: no-cache`
  - `no-cache`

Response Caching Example

The relationship between global, mashable-type and artifact level configuration for response caching provides a wealth of control, but sometimes can be confusing. This example illustrates some common configuration patterns and behavior.

	Event	Cache Entry
1.	UserA turns response caching on for a spreadsheet mashable he just registered.	The caching configuration is saved, but no caching will occur because response caching has not been enabled.
2.	A MashZone NextGen administrator turns response caching on in the Admin Console.	Response caching is now allowed but can occur only for the spreadsheet mashable configured in step 1. **Note:** If no caching configuration had been set in step 1, no response caching would occur after this step as both are required for caching to occur.
3.	A MashZone NextGen administrator turns caching on for all RSS and Atom mashables, but does not configure an expiration period. UserA adds a block to a mashup in Wires for the Yahoo Financial RSS mashable and previews the results.	The response from Yahoo is cached with an expiration of 5 minutes, taken from the global default expiration period.
4.	The owner of the Yahoo Financial RSS mashable changes the response cache expiration period for this mashable to 1 minute.	The existing response cache entry for Yahoo Financial is used as the input to the filter block. This cache entry will remain for the full 5 minutes and then expire.

	Event	Cache Entry
	Within seconds UserA adds a filter block to his mashup, connects the Yahoo Financial block and previews the result of the filter.	All subsequent responses for the Yahoo Financial mashable will use a 1 minute expiration period.
5.	UserA finishes his mashup and creates a basic app with it. He turns response caching on for his mashup and assigns an expiration period of 15 minutes. UserB uses this app. Three minutes later, UserC also uses this app.	The mashup is run and caches its response when UserB runs the associated app. The mashup also runs the Yahoo Financial mashable and that response is also cached. When UserC runs the app three minutes later, the cache entry for Yahoo Financial has expired. The app, however, simply uses the cached mashup response which is based on Yahoo Financial results from three minutes before that may now be considered stale.
6.	A MashZone NextGen administrator sets the default response caching expiration period for all RSS and Atom mashables to 3 minutes.	As RSS or Atom mashables are run, their responses are cached for 3 minutes, *except* for Yahoo Financial which is cached for 1 minute.
7.	UserA registers a REST mashable with several operations for the Human Resources system in his organization. To ensure that sensitive information is not cached, he enables caching for the REST mashable as a whole but turns off response caching for two sensitive operations.	When users work with the REST mashable, or any mashup or app based on this mashable, responses will be cached except for those operations where caching has been disabled.
8.	A MashZone NextGen developer creates a mashup that uses several Atom mashables. In the mashup he adds the HTTP Cache-Control header with a value of `no-cache` to the response of each of the Atom mashable invocations.	When this mashup is used, each Atom mashable will be invoked every single time with no response caching because of the HTTP response headers.

Manage Data Sources and Drivers

Data sources combine the connection and driver information needed to work with both the MashZone NextGen Repository and with other databases. Data sources can use either JDBC connections or a JNDI connection pool. Once you have created a data source, you and other users can use it as the source for any number of mashables or in mashups using EMML or for dashboards.

See "Add a Data Source" on page 1679, "Edit, Test or Remove Data Sources" on page 1681 and "Add or Manage JDBC Drivers" on page 1681 for instructions.

Add a Data Source

if you use connection pools to connect to databases, configure JNDI in your application server to enable access to the connection pools as needed.

Procedure

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **JDBC Configuration** tab.
3. If needed, add the JDBC driver for this database to MashZone NextGen. See "Add or Manage JDBC Drivers" on page 1681 for instructions.
4. Click **Data Sources** to see a list of existing data sources.

   Initially, this lists the data source for the MashZone NextGen Repository and for the Snapshots repository.
5. Click **Add data source** to create a new data source.
6. Enter a **Data source Name** for a new data source.

   Data source names may contain ASCII alphabetic characters and numbers *only*. Data source names may *not* contain any punctuation or symbols, such as periods (.), dashes (-) or underscores (_).
7. Select the appropriate driver for this datasource in the **JDBC Driver** drop-down menu.
8. In the **JDBC URL** field, enter either the URL for a JDBC connection or the JNDI name for a connection pool to connect to this data source. Common URL or JNDI forms include:
   - `jdbc:mysql://`*hostname*`/`*databasename*

     Important: For MySQL databases, it is *recommended* that you include the database name in data source URLs. If this information is omitted, testing the data source fails and may also cause errors with access to stored procedures.
   - `jdbc:oracle:`*driver-type*`@`*hostname*`:`*port*

- `jdbc:postgresql://`*hostname* `:`*port* `/`*database-name*
- `jdbc:jtds:sqlserver://`*hostname* `:`*port* `;`*database-name*
- `jdbc:sqlserver://`*hostname* `:`*port* `;databaseName=`*database-name*
- `jdbc:sybase:Tds:`*hostname* `:`*port*
- `java:`*context-path* `/`*jndi-resource-name* or *context-path* `/`*jndi-resource-name*

9. Optionally, enter the **Username** and **Password** to use to connect to this database.
10. Click **Show connection pooling options** to display further options.
11. Optionally, set connection pooling options for this data source:
    - **Initial Size** = the initial number of connections to create when the pool for this data source starts up. This defaults to 0.
    - **Maximum Active** = the maximum number of connections that can be allocated at one time for this data source. This defaults to 8. Set this to -1 to remove all limits.
    - **Maximum Wait** = the maximum number of milliseconds that the pool will wait when no connections are available before failing. Defaults to -1 which is an indefinite wait.
    - **Maximum Idle** = the maximum number of connections that can be idle without connections being released for this data source. Defaults to 8. Set this to -1 to prevent any connections being released.
    - **Minimum Idle** = the minimum number of idle connections that can exist before new connections are added to the pool for this data source. This defaults to 0, indicating no new connections should be created.
    - **Pool Prepared Statement** = set this option to allow prepared statements for the database mashables that use this datasource to be pooled. This is disabled by default.

      The usefullness and effect of pooling prepared statements depends on the type of database for this connection. See documentation for your database for more information or recommendations.
    - **Validation Query** = the SQL query that is used to validate connections in the pool for this datasource.
    - **Validation Call Timeout** = the number of milliseconds before a connection validation check is considered to have failed, causing the pool to invalidate and discard the connection. If you set this property to a number less than zero, validation calls do not expire, which is the default behavior.
    - **Time Between Eviction Runs** = the number of milliseconds between tests for idle connections for this datasource. This defaults to -1, which prevents all idle connection testing.
    - **No of tests per run** = the number of connections to test during any idle test run for this datasource. This defaults to 3.

- **Minimum Evictable Idle Time** = the minimum number of milliseconds that a connection can be idle before being tested for eviction. This defaults to 30 minutes (1800000 milliseconds).

> **Note:** For more details on connection pooling properties, see Apache DBCP Documentation

12. Click **Save changes**.

Edit, Test or Remove Data Sources

To manage data sources

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **JDBC Configuration** tab.
3. Click **Data Sources** to see a list of existing data sources.

   Initially, this lists the data source for the MashZone NextGen Repository and for the Snapshots repository.
4. To edit a data source, click the **Edit** icon on the line for that data source and change properties.

   See "Add a Data Source" on page 1679 for information on specific data source properties.
5. To test the connection to a data source, click the **Test** icon on the line for that data source.
6. To delete a data source, click the **Delete** icon on the line for that data source.

> **Important:** Do *not* delete the data source for either the MashZone NextGen Repository or the Snapshots repository.

Add or Manage JDBC Drivers

To manage JDBC drivers

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **JDBC Configuration** tab.
3. Click **JDBC Drivers**.

   A list of existing drivers displays. This initially contains just the driver for the default MashZone NextGen Repository.
4. To remove a driver, click **Delete** on the line for that driver.

> **Important:** Do *not* delete the driver for the MashZone NextGen Repository.

5. To add a new driver:
   a. Click **Add new JDBC driver**.
   b. Enter a **Name** for a new driver.
   c. Enter the Java **Class** name for this driver.
   d. Click **Browse** and find the JAR file for this driver.
   e. Click **Add this JDBC driver**.

Upgrade JDBC connections

With the MashZone NextGen version 9.10 release the persistence of JDBC drivers and connections have been changed. And only one type of JDBC connections is still available. With this version MashZone NextGen supports the import of JDBC connections from MashZone and Presto legacy versions.

Upgrade JDBC connections of MashZone versions 9.5 to 9.9

Procedure

1. Copy all drivers located in the jdbcdrivers folder of the MashZone installation directory to the jdbcdrivers folder of the MashZone NextGen installation and restart the MashZone NextGen server.
2. Export in MashZone all JDBC connections you want to upgrade. See MashZone Online help for details.

   The exported JDBC connections are stored as mzp files, starting with A_DATABASE…, in folder <*MashZone installation*> \ppmmashzone\server\bin \work\work_mashzone_m\mashzone_data\importexport.
3. Copy all mzp files starting with A_DATABASE... into the dbconnections subfolder of the importexport folder located in <MashZone NextGen installation>\Presto \mashzone\data\importexport.
4. Start the MashZone NextGen server if not already done. Then go to the runtool folder (located under Presto\mashzone\tools), open a dos command line and call:

   ```
   importtool -user Administrator -password manage -folder dbconnections
   ```

All JDBC connections from the dbconnections folder will be imported into MashZone NextGen.

In the MashZone NextGen Admin Console the JDBC connections are separated in two parts, the driver part and the data source part. You can find all JDBC related items in the **JDBC Configuration** tab of the Admin Console.

If you need to upgrade a connection using a JDBC driver that consists of multiple JAR files, you will have to create a new driver JAR which bundles all the individual files into one single file. After that, you will have to copy the newly created JAR file to the MashZone NextGen installation.

Upgrade JDBC connections of Presto version 3.9 and 9.9

Procedure

This upgrade is relevant for MashZone legacy JDBC connections.

1. Copy all drivers located in the jdbcdrivers folder of your Presto installation into thejdbcdrivers folder of the MashZone NextGen installation and restart the MashZone NextGen Server
2. Check if the **MashZone** tab exists in the Admin Console of Presto. If not, open thepresto.config file located in <*Presto installation*> \apache-tomee-jaxrs\webapps\presto\WEB-INF\classes\ and set the `mashzone.administration.disabled=false` flag and restart the Presto server.
3. Export the required connections in the Presto Admin Console (**Admin Console** -> **MashZone** -> **Database Connections** tab. You have to export each connection separately. See Presto online help for details.

   The exported JDBC connections are stored as mzp files, starting with A_DATABASE…, in the importexport folder of your Presto installation.
4. Copy all database related mzp files in the **importexport** folder of your Presto installation to the dbconnections subfolder of the importexport folder in your MashZone NextGen installation.
5. Start the MashZone NextGen server if not already done. Then go to the runtool folder (located under Presto\mashzone\tools), open a dos command line and call:

   ```
   importtool -user Administrator -password manage -folder dbconnections
   ```

All JDBC connections from the dbconnections folder will be imported into MashZone NextGen.

In the MashZone NextGen Admin Console the JDBC connections are separated in two parts, the driver part and the data source part. You can find all JDBC related items in the **JDBC Configuration** tab of the Admin Console.

If you need to upgrade a connection using a JDBC driver that consists of multiple JAR files, you will have to create a new driver JAR which bundles all the individual files into one single file. After that, you will have to copy the newly created JAR file to the MashZone NextGen installation.

Upgrade JDBC configuration of Presto version 3.9 and 9.9

Procedure

1. To export all existing JDBC configurations from Presto version 3.9 and 9.9 go to the \prestocli\bin folder of the Presto installation, open a dos command line and call:

   ```
   pAdmin exportDatasource -l http://localhost:8080/presto/esd/api -f
   JDBCConnections_backup.zip -u Administrator -w manage -j
   ```

   The created zip file contains all information about JDBC configurations of Presto, including the related drivers.

2. Copy the JDBCConnections_backup.zip file to the prestocli\bin folder in your MashZone NextGen installation.
3. Go to prestocli\bin folder in your MashZone NextGen installation, open a dos command line and call:

```
pAdmin importDatasource -f JDBCConnections_backup.zip -u
Administrator -w manage -o
```

All Presto JDBC configurations will be imported into MashZone NextGen.

Configure the Default Operations Generated for Database Mashable

When users or MashZone NextGen administrators register simple or custom mashables for databases, MashZone NextGen generates a default set of operations for the tables, views or stored procedures in the selected database. See "Default Operations Available for Database Mashables" on page 118 for details.

For simple database mashables, users have no control over which operations are generated. MashZone NextGen administrators can choose which of the default operations are generated for custom database mashables.

Some of these operations have security implications because they allow users to define portions of the SQL statements dynamically. Other operations allow users to insert, update or delete records in tables. Some types of queries have performance implications as they can have excessively large result sets.

In addition to database mashables, mashups can also directly update databases using the <sqlUpdate> statement in EMML or the SQL block in Wires.

You can manage which operations are generated by default for database mashables in the Admin Console.

To configure default operations

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Mashable Database Services** section and click **Service Generator Settings**.
3. Change any of the options for Unsecure Operations:
   - *Exclude All Unsecure Operations* = this option determines whether users can choose to include operations in database mashables that do not use prepared statements and thus pose a risk of SQL injection attacks.

     This is enabled by default. It ensures that the `select`*`table-name`*, `find`*`table-name`*`Where` and `find`*`table-name`*`ByWhereClause` operations are not included in any database mashable when the mashable is registered.

     If you clear this option, the default availability of these operations is determined by the *Include selectTable operations* and *Include findTableWhereColumn operations* options.

- *Include findTableByWhere operations* = this option determines the default availability of the `findtable-nameByWhereClause` operation for tables in a given database mashable. See "Arbitrary SQL Queries for Database Mashables" on page 124 for more information on the uses and effects of this operation.

  This option is enabled, by default.

- *Include selectTable operations* = this option determines the default availability of the `selecttable-name` operation for tables in a given database mashable. This operation does not use prepared statements and thus is a potential security risk for SQL injection attacks.

  This option is disabled by default. MashZone NextGen administrators can choose, however, to include this operation for custom database mashables unless the *Exclude All Unsecure Operations* option is set.

  Set this option to include the `selecttable-name` operation by default.

4. Change any of the options for Large Queries:

   - *Include findAllTable operations* = this option determines the default availability of the `findAlltable-name` operation for tables in a given database mashable. This is true, by default.

     These types of options can have really large result sets and thus can have a performance impact.

   - *Include findTableWhereColumn operations* = this option determines the default availability of the `findtable-nameWhere` operation for tables in a given database mashable. This operation does not use prepared statements and thus is a potential security risk for SQL injection attacks.

     This flag is false, by default, but MashZone NextGen administrators can choose to include this operation for custom database mashables unless the *Exclude All Unsecure Operations* option is set.

5. Change any options for Table Updates:

   - *Include insert operations* = determines the default availability for `inserttable-name` operations for database mashables which insert records in database tables. This is set by default.

   - *Include update operations* = determines the default availability for `updatetable-name` operations for database mashables which update records in database tables. This is set by default.

   - *Include delete operations* = determines the default availability for `deletetable-name` operations for database mashables which delete records in database tables. This is set by default.

   Note: These options also affect the use of the <sqlUpdate> statement in EMML and the SQL block in Wires for mashups.

6. Click **Save Settings**.

Manage Categories for Mashups, Mashables and Apps

Categories allow you to define the top level categories for the purpose or focus of mashable information shources, mashups or apps. Categories answer the question "what is this artifact use for? what is it about?." User tags can then further refine th subject or puprose or aftifacts.

To manage categories

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Taxonomies** section and click **Categories**.

   A list of existing categories displays. To delete an existing category, click **Delete** on the line for that category.
3. Click **Add New Category** and complete these properties:
   - **Name** = the name for this category. Category names may contain letters and numbers only. Symbols or punctuation are not allowed.
   - **Description** = a short description of this category.
4. Click **Save**.

Manage Providers for Mashups, Mashables and Apps

Providers identify the organizations that provide mashable information sources, mashups or apps in MashZone NextGen. Users can search for artifacts based on providers. Provider information also helps users determine if an artifact is one they trust or are interested in using.

If you have installed sample mashables or mashups provided with MashZone NextGen, providers are defined for these sample artifacts.

In addition to defining providers for partners or other organizations that provide source information, it is a best practice to define at least one provider for your own organization so that users may assign this to artifacts as they create them. It may be more useful to define providers for the different departments or groups in your organization to enable finer grained information.

To manage providers

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Taxonomies** section and click **Providers**.

   A list of existing providers displays.
   - To delete an existing provider, click **Delete** on the line for that provider.
   - To edit an existing provider, click **Edit** on the line for that provider.

3. Click **Add new provider** to add a new provider and complete these properties:
   - **Name** = the name of the organization or group for this provider.
   - **Description** = a short description of this provider.
   - **URL** = the URL to the web site or as an identifier for this provider.
4. Click **Save**.

Work With MashZone NextGen Attributes

You use MashZone NextGen attributes to provide credentials or other runtime inputs for apps, mashables or mashups.You can also use attributes to provide additional meta data for artifacts. Common examples include information for the current user, such as a credential for a mashable, shared information that should be used for all contexts, such as an application ID that should always be used for a specific mashable, or information from a previous response that is specific to the current transaction or session.

MashZone NextGen attributes can be defined for these contexts:

- **User Attributes**: are defined by each user as part of their profile in MashZone NextGen and saved in the MashZone NextGen User Repository. See "Manage Your MashZone NextGen User Attributes" on page 82 for more information.

  You can also choose to expose attributes from the MashZone NextGen User Repository as MashZone NextGen user attributes. You can expose attributes from the default User Repository or from your LDAP Directory, if MashZone NextGen is configured to use LDAP. See "Expose User Attributes from the User Repository in MashZone NextGen" on page 1689 for more information.

  At runtime, the MashZone NextGen Server first checks the MashZone NextGen user attributes for the current user to resolve any references to MashZone NextGen attributes used in a request. If there is no matching user attribute for the current user, the MashZone NextGen Server uses the matching MashZone NextGen global attribute to resolve the attribute value before processing the request.
- **Global Attributes**: can be used to define a default or shared value for a MashZone NextGen user attribute. This can be used for all users or just for those users that have not defined a specific value.

  Global attributes can also be used to define any common value that should be used in many contexts. See "Manage MashZone NextGen Global Attributes" on page 1688 for instructions on creating global attributes.
- **Session Attributes**: are data that is available for one user session. They can be defined or updated in these ways:
  - **From data in a mashable or mashup response**. These MashZone NextGen session attributes are defined in requests using a MashZone NextGen Header/Parameter. For more information, see "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502.

  - **In mashup scripts**. See "Using MashZone NextGen Attributes in Mashups" on page 614 for more information.
- **Request Attributes**: can be used to refer to HTTP headers or cookies for a specific request. For more information, see "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502.
- **Artifact Attributes**: are additional meta data that you provide for individual apps, mashables or mashups. See "Manage Artifact Attributes" on page 1690 for more information.

Manage MashZone NextGen Global Attributes

MashZone NextGen global attributes define:

- Default or common values that should be used for input parameters to mashables, mashups or apps.
- Default or shared values for MashZone NextGen user attributes. For example, global attributes can define a credential that many users can share to invoke a mashable. Some users may have their own credentials which are defined as a MashZone NextGen user attribute of the same name.
- Credentials or other configuration used in connections to SharePoint when your site has the MashZone NextGen Add-On for SharePoint Add-On. For more information on global attributes used with SharePoint connections, see configuring "Configure MashZone NextGen Connections to SharePoint" on page 1655.
- Schemas for datasets used in RAQL queries in MashZone NextGen Analytics. See "Global Dataset Schemas as MashZone NextGen Attributes" on page 725 for more information.

To manage MashZone NextGen global attributes

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Open the **Attributes** section and click **Global Attributes**.
3. To add a global attribute:
   a. Click **Add new global attribute**.
   b. Enter a **Name** for the attribute.

      MashZone NextGen attributes can have any name, but they must be unique within a given MashZone NextGen Repository. If a global attribute is to be used as a default for a MashZone NextGen user attribute, then the attribute names must match.
   c. Enter the **Value** for the attribute.
   d. If the value of the attribute should be encrypted, such as for a password, set the **Encrypted** option.

The value of the attribute is encrypted in the MashZone NextGen Repository and will automatically be decrypted when it is used.

e. Click **Save**.

4. To update a global attribute, click **Edit** for that attribute in the list and update the name or value.
5. To delete a global parameter, click **Delete** for that attribute in the list.
6. Restart the MashZone NextGen Server to apply these changes. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Expose User Attributes from the User Repository in MashZone NextGen

You can access specific user attributes from the User Repository when you run apps, mashables or mashup using MashZone NextGen user attributes. Typically, you use MashZone NextGen attributes (user, global or session) as tokens for the values for input parameters. The MashZone NextGen Server resolves these tokens when the artifact is run. See "Map MashZone NextGen Attributes to Artifact Input Parameters" on page 1502 for the basic syntax to use MashZone NextGen user attributes.

The specific attributes from your user repository that are visible as MashZone NextGen user attributes depends on how you have configured the User Repository in MashZone NextGen:

Default User Repository	All repository attributes are accessible. This includes: ■ `firstName` ■ `lastName` ■ `email` ■ `username` ■ `password`
LDAP Directory	The LDAP attributes that are visible as MashZone NextGen user attributes is determined by the configuration you set when you integrate LDAP with MashZone NextGen. By default, this includes: ■ `uid` ■ `givenName` ■ `sn` ■ `mail` ■ `cn` ■ `postalCode`

	See "Integrate Your LDAP Directory with MashZone NextGen" on page 1590 for more information.

Manage Artifact Attributes

Artifact attributes provide additional meta data about individual apps, mashups or mashables that can be used for more sophisticated or complex requirements when you use MashZone NextGen in your organization. Artifact attributes are also sometimes use to provide overrides to default information, such as the expected character encoding for mashable responses.

To use custom artifact attributes, you must define the attribute for the different types of artifacts it should be used with. See "Add an Artifact Attribute Definition" on page 1690 for instructions. You or other artifact owners can then set the values for these attributes. See "Add or Update Custom Attributes" on page 88 for instructions.

For more information, please contact your Software AG sales representative.

Add an Artifact Attribute Definition

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Open the **Attributes** section and click **Attribute Definitions**.
3. Choose the type of artifact (mashable, mashup or app) that this attribute should apply to.
4. Enter an **Attribute Name**.

   This must be a unique name in this MashZone NextGen Repository. Names should be limited to alphanumeric characters (letters and numbers) and the underscord (_) character.
5. Choose the `Enum` datatype if you wish to provide a list of valid values for users to select for this attribute. Choose `String` if users should enter a value for the attribute.
6. If this is an `Enum` datatype, enter the list of values that users should select from as a comma-separated list in **Possible values**. Enter values in the order in which you wish them to appear.
7. Optionally, enter a default value for this attribute.
8. Click **Add this artifact attribute**.

   Once this is saved, users will see a field for this attribute in the Info tab for every artifact of this type.

Edit or Delete Artifact Attribute Definitions

You can edit or delete artifact attribute definitions in the Admin Console. Open the **Attributes** section and click **Attribute Definitions**. The click **Edit** or **Delete** for that attribute in the list that displays.

Deleting an artifact attribute ensures that this information no longer displays in MashZone NextGen for artifacts of that type. It does *not* remove any values assigned to artifacts in MashZone NextGen for that attribute.

Disable Mashup Features

Some features in EMML have security or performance implications that you may need to manage. You can disable or re-enable these mashup features in the MashZone NextGen Server:

To enable/disable mashup features

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Platform Features** section and click **Mashup Engine**.
3. Set or clear the following properties:
   - *Enable Scripting* = this feature is the <script> statement in EMML which allows mashups to call JavaScript or JRuby scripts. This also provides direct access to Java language classes.

     This feature is enabled by default. Scripting can have security implications and also has a performance impact. Clear this option to disable scripting.
   - *Enable Direct SQL* = this feature is the <sql> and <sqlUpdate> statements in EMML. These statements allow mashups to issue raw SQL statements to databases which can have security implications.

     This feature is enabled by default. Clear this option to disable direct SQL statements in mashups.
   - *Enable EMML Profiling* = this option determines whether performance profiling for mashups is enabled. Performance profiling allows developers to see how long processing took for each statement in their mashups when they test them. Profiling does, however, have a performance impact.

     Performance profiling is disabled by default. Set this option to enable performance profiling.
   - *Enable direct invocation* = this feature is the <directinvoke> statement in EMML. This statement allows mashups to invoke web services or other information sources from any URL without authentication or authorization by MashZone NextGen.

     This feature is enabled by default. Clear this option to prevent direct invocation of ungoverned information sources.
4. Click **Save**.

Configure HTTP Response Header Forwarding

This topic is valid for MashZone NextGen 3.1 and above.

When mashable information sources or mashups are invoked in MashZone NextGen, the MashZone NextGen Server does not necessarily include any of the standard HTTP response headers from the mashable or mashup in its own response. The HTTP response whitelist property for the MashZone NextGen Server defines those standard HTTP response headers from mashable or mashup responses that should automatically be included in MashZone NextGen Server responses.

Note: Unless HTTP response headers have been completely disabled (see below), *all* custom HTTP headers beginning with `X-` or `x-` are automatically forwarded from the mashable or mashup response.

Specific requests to invoke mashables may also override the default header forwarding setting using "MashZone NextGen Headers/Parameters" on page 1511.

You can add HTTP headers to the header whitelist. You can also completely disable all HTTP response headers.

To manage HTTP response headers

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Platform Features section and click **HTTP Processing**.
3. To disable *all* HTTP response headers for mashables or mashups, clear the **Include HTTP headers** option.
4. Add the names of the standard headers that should be forwarded to the end of the **Whitelist** property in HTTP Response Processing.

   Note: Be sure to separate each header name with a comma. Do *not* remove any of the following headers that are included in the whitelist by default:

   - Cache-Control
   - Content-Disposition
   - Content-Type
   - Date
   - Etag
   - Expires
   - Last-modified
   - serviceURL
   - Set-Cookie

5. Click **Save settings** (in the HTTP Response Processing section).

Configure Mashable HTTP Request Timeouts

This topic is specific to MashZone NextGen 3.1.

MashZone NextGen uses the default HTTP handling for requests to mashable information sources. This does not set any timeout period for a connection to be made or a timeout for a response to be received.

In most cases, this default behavior is appropriate. If denial of service attacks are a concern, however, this default behavior may not be acceptable.

You can set timeout periods for requests to all mashable information sources that use HTTP. This affects Atom, RSS, REST (including remote XML and CSV files), SharePoint, remote Spreadsheets and WSDL mashables. Database mashables and local spreadsheet, XML and CSV mashables are not affected.

To manage HTTP request timeouts

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Platform Features section and click **HTTP Processing**.
3. In the HTTP Request Processing section:
   - Enter a number of milliseconds in **Connection Timeout** to define the timeout period for connections to be successfully made with mashable information sources.
   - Enter a number of milliseconds in **Socket Timeout** to define the timeout period for a response to be received from mashable information sources.
4. Click **Save settings** (in the HTTP Request Processing section).

Enable or Disable the Snapshot Feature

The snapshots feature allows users to save the specific results when a mashable or mashup is run and use this data in mashups or apps. Users or administrators can also schedule snapshot jobs to take snapshot automatically.

This feature is disabled by default when you install MashZone NextGen.

To disable/enable the snapshot feature

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Platform Features** section and click **Snapshots**.
3. Set the **Enable Snapshots** option.

Set Web Feed Normalization

By default, MashZone NextGen returns RSS and Atom web feed responses in the format received from the syndicated web feed. In most cases, this ensures that all the information from the web feed is returned. But it can make combining results from different web feeds difficult.

The default setting (`native`) disables *normalization*. You can choose to enable normalization to have the MashZone NextGen Server automatically convert the results for either all RSS or all Atom feeds to one specific standard and version.

Note: If you enable normalization, there are specific issues and work-arounds. See "Normalization and MashZone NextGen Support for RSS/Atom Formats" on page 132 for details. MashZone NextGen developers can also override normalization configuration for specific invocations of a web feed using "MashZone NextGen Headers/Parameters" on page 1511.

To set web feed normalization

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. If needed, expand the **Server** section.
3. Click **Syndication Response**.
4. Choose:
   - One of the RSS versions to normalize all web feed to that version of the RSS standard.
   - One of the Atom versions to normalize all web feed to that version of the Atom standard.
   - `native` to disable normalization.

Handle SOAP Encoding Errors for WSDL Services

Some WSDLs for web services use array datatypes from the SOAP Encoding namespace, `http://schemas.xmlsoap.org/soap/encoding`, however MashZone NextGen does not automatically provide this schema for WSDL web services. This can cause unknown type or type resolution errors during registration of WSDL Web Services. To fix these errors, you must add a configuration property to the MashZone NextGen Server.

To add a configuration property

1. In any simple text editor, open the presto.config file in the *MashZoneNG-config* folder.

   Note: The *MashZoneNG-config* folder may be an external folder outside the MashZone NextGen Server or it may be in the default locations. See

"Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information on possible locations.

2. Add `nsd.compile.schemas=http://schemas.xmlsoap.org/soap/encoding` on a new line in this file and save this change.
3. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Add XML Schemas to the Wires Mapper Block

The Mapper block in Wires allows users to transform results in a mashup to a well-known structure defined in an XML schema. These schemas may include public standards, such as RSS or Atom, or schemas used in mashable information sources or systems in use within your organization.

Note: Wires supports only XML schemas (*.xsd files). It does not support DTDs or RELAXNG grammars.

The Mapper block can map to the structure defined by an existing block in a mashup. It is generally more convenient, however, to upload the XML schemas most commonly in use in your organization.

To add a schema

1. If the schema you choose to upload imports other schemas using `<xs:import>` statements, you *must first*:
   - Upload each of the imported schemas before you upload the schema that imports them.
   - Update the path in the `<xs:import>` statements in the importing schema to use a relative path with just the imported schema file name.
2. Click Admin Console in the MashZone NextGen Hub main menu.
3. Expand the Platform Features section and click **Wires Schema**.

   This lists any schemas that have already been upload.
4. Click **Add new schema**.
5. Click **Browse** to find and select the XML schema you want to use.
6. Enter a name or general description and click **Add Schema**.

The schema is now available in the Mapper block.

You can use the list of schemas in this page to delete schemas. Or, you can find and mange the schema file, named /system/Wires/schemas/*xsd-file-name* , using **File Resources** in the Platform Features section of the Admin Console. See "Manage Files for MashZone NextGen Features or Artifacts" on page 1745 for more information.

Integrated MashZone Server Configuration and Administration

MashZone NextGen uses the Integrated MashZone Server to manage and process MashZone feeds that you import into MashZone NextGen or that users create in MashZone NextGen, if new MashZone feeds are allowed. Connection information is required before administrators or users can import, edit or create MashZone feeds.

MashZone feeds are similar to MashZone NextGen mashables and mashups in that they access information from a variety of sources. For MashZone NextGen, MashZone feeds also allow you to access data from Optimize or Process Performance Manager.

To integrate the MashZone Server with MashZone NextGen, you must first:

- Move the MashZone Repository to a robust solution. See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for instructions.
- Make the MashZone Feed Editor available in the MashZone NextGen Hub. See "Enable the MashZone Feed Editor" on page 1703.
- Make the MashZone Administration available in the MashZone NextGen Hub. See "Enable the MashZone Administration" on page 1704.
- Set basic configuration for the Integrated MashZone Server, such as proxy server information. See "Basic Settings for the Integrated MashZone Server" on page 1696 for instructions.
- "Tune Memory/Caching for the Integrated MashZone Server" on page 1697
- Import existing MashZone feeds into MashZone NextGen.

  See "Import or Export MashZone Feeds" on page 1701 for instructions.
- "Event Service Configuration and Administration" on page 1704 The Integrated *Event Service* allows MashZone NextGen to create and handle *event mashables* that connect to the Event Bus andApama , subscribe to specific event types and then use data from published events as an information source.
- Integrate with Process Performance Manager (PPM). This configuration allows users to work with the PPM Chart app to find and include PPM charts in workspace apps or to define data sources for MashZone feeds from PPM.

  See "Process Performance Manager (PPM) Integration" on page 1732 for more information and links.

Basic Settings for the Integrated MashZone Server

Once you have defined the connection to the Integrated MashZone Server, you can complete basic settings that allow you to:

- Create New MashZone Resource Directories

  Data sources used in MashZone feeds that are file based, such as Excel spreadsheets, CSV files or XML files, must be stored in resource directories known to the Integrated MashZone Server. Resource directories also allow you to restrict access to data sources to specific users or user groups.

- Manage MashZone Resource Directories

  You can update, delete, share, export and import resource directories.

- Configure Proxy Information for the Integrated MashZone Server

- Enable the Geocoding Operator for MashZone Feeds

  This operator uses the Google Maps API. To enable the operator, you must accept license terms and optionally provide an API key.

Tune Memory/Caching for the Integrated MashZone Server

MashZone NextGen, the Integrated MashZone Server and the Event Service share the local Java heap memory. Heap memory is also used for internal caches and MashZone NextGenIn-Memory Stores used in MashZone NextGen Analytics. MashZone NextGen can also be configured to use off-heap memory if you have installed BigMemory Servers. For more information, see "Memory Configuration for the MashZone NextGen Server" on page 1627and "About BigMemory and the MashZone NextGen Analytics In-Memory Stores" on page 861.

The Integrated MashZone Server and the Event Service are initially installed based on assumptions for a small web application. This default memory allocation may work well for development environments, but may need to be adjusted for staging or production environments. Memory requirements for MashZone NextGen, the MashZone NextGenIn-Memory Stores and event sources in the Event Service may also affect the overall available memory, requiring tuning for MashZone internal caches.

You may adjust configuration for both Java heap memory and memory configuration for the internal caches used by the Integrated MashZone Server using the following techniques:

- Tune MashZone Memory and Cache Configuration Manually.

  Manual tuning gives you greater control to balance memory requirements for MashZone NextGen, the Integrated MashZone Server and the Event Service, but does require manual updates to several configuration files.

- Automatically Tune MashZone Memory and Cache Configuration

  This uses a simple script to automatically update memory and cache configuration based on preset sizes. These preset values, however, do not take any memory requirements for MashZone NextGen or MashZone NextGenIn-Memory Stores into account and thus may not be suitable in some circumstances.

Tune MashZone Memory and Cache Configuration Manually

To manually update memory and cache configuration

1. Update Java Heap Settings
2. Update Cache Memory Settings
3. Update MashZone ThreadSize Properties.
4. Then restart the MashZone NextGen Server to apply this change. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

Update Java Heap Settings

To update the size of the Java Heap

1. In a text editor of your choice, open the setenv.bat or setenv.sh file, based on your operating system, in the *MashZoneNG-install* /apache-tomee-jaxrs/bin/ folder.
2. At a minimum, update the `-Xmx` value to tune the maximum size of heap. See the preset suggestions in Automatically Tune MashZone Memory and Cache Configuration as starting points and Memory Configuration for the MashZone NextGen Server for suggestions for MashZone NextGen.
3. Save your changes.

Update Cache Memory Settings

1. In the text editor of your choice, open the ehcache.xml file in the *web-apps-home* / mashzone/WEB-INF/classes folder.
2. Update the `maxBytesLocalHeap` value on the `<cache>` elements with the following names:
   - `RESULT_FEED_BASE`
   - `RESULT_FEED_TOP`
   - `RESULT_FEED_DEBUG`

   See the preset suggestions in Automatically Tune MashZone Memory and Cache Configuration as starting points.
3. Save your changes

Update MashZone ThreadSize Properties

1. In the text editor of your choice, open the mashzone.properties file in the *web-apps-home* /mashzone/WEB-INF folder.
2. Update the following properties:
   - `calculation.threadpool.coresize`

- `calculation.threadpool.maxsize`

  See the preset suggestions in Automatically Tune MashZone Memory and Cache Configuration as starting points.

3. Save your changes

Automatically Tune MashZone Memory and Cache Configuration

This uses a simple script to automatically update memory and cache configuration based on preset sizes. These preset values, however, do not take any memory requirements for MashZone NextGen or MashZone NextGenIn-Memory Stores into account and thus may not be suitable in some circumstances.

1. Determine which present configuration you want to use.

   Preset memory configuration is defined by three sizes: *s* = small, *m* = medium and *l* = large. These presets are configured based on the following assumptions:

Preset Option	Heap	Core Threadsize	Maximum Threadsize	Internal Caches
S: a small application on a host with: ■ 64 bit ■ 2 Cores ■ 4G of memory	1G	4	4	■ Base = 125M ■ Top = 100M ■ Debug =25M
M: a medium application on a host with: ■ 64 bit ■ 4 Cores ■ 16G of memory	8G	8	12	■ Base = 1G ■ Top = 800M ■ Debug =200M
L: a large application on a host with: ■ 64 bit ■ 8 Cores ■ 64G of memory	16G	16	24	■ Base = 2G ■ Top = 1.6G ■ Debug = 400M

2. Open a command or terminal window and move to the *MashZoneNG-install* /mashzone/tool/runtool folder.
3. Enter the appropriate command shown below based on your operating system:
   - For Windows, enter `upgradetool.bat -system preset-size`

     Using the size option you determined in step 1.
   - For Linux, OS/X or UNIX, enter `upgradetool.bat -system preset-size`

     Using the size option you determined in step 1.
4. If needed, manually increase the Java heap allocation to accommodate both MashZone NextGen and the Integrated MashZone Server. See "Memory Configuration for the MashZone NextGen Server" on page 1627 for instructions.

Create New MashZone Resource Directories

To work with data sources in MashZone feeds that are file-based, such as Excel spreadsheets, CSV files or XML files, you must store the files in a *resource directory* that the Integrated MashZone Server knows. This can be the default resource directory:

MashZoneNG-install /mashzone/data/resources

Or it can be any subdirectory of the default.

You can also use resource directories to control access to data source files to specific users or groups.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. See the MashZone documentation for the remaining steps.

Manage MashZone Resource Directories

Resource directories hold file-based data sources, such as Excel spreadsheets, CSV or XML files. MashZone NextGen administrators can:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. Consult the MashZone documentation to perform the following:
   - Edit a Resource Directory
   - Delete a Resource Directory

- Share Resource Directories to determine which users and groups can work with these data sources.
- Import Resource Directories
- Export Resource Directories

Configure Proxy Information for the Integrated MashZone Server

If a proxy server is used to access data sources for MashZone feeds, you must add configuration information for the proxy server.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. See the information on Set proxy server settings in the MashZone documentation for the remaining steps.

Enable the Geocoding Operator for MashZone Feeds

This operator for MashZone feeds uses The Google Maps APIs. To use this operator, you must accept the Google license terms.

You may also need to enable unrestricted access with a Google Maps API Premier key to support cumulative transactions greater than 2,500 per day for the MashZone feeds that use this operator.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. See the information on entering Google Maps key in the MashZone documentation for the remaining steps.

Import or Export MashZone Feeds

You can import existing MashZone feeds from your existing MashZone Servers into MashZone NextGen to use them to create apps or workspace apps. You can also export any or all MashZone feeds registered in MashZone NextGen and the Integrated MashZone Server.

Import Existing MashZone Feeds

If you have data feeds defined in an existing MashZone Server, you can import them into MashZone NextGen to use them in apps or workspace apps.

1. Export the MashZone data feeds that you want from your existing MashZone Server and *omit* permissions. See MashZone help for instructions on completing this step.

   Important: Do not export MashZone feeds that use connections to Terracotta BigMemory caches as a data source. These connections are not supported by the Integrated MashZone Server for MashZone NextGen.

   Instead, connect to BigMemory caches using MashZone NextGen Analytics. See "MashZone NextGen Analytics" on page 685 for links to more information.

   This creates an MZP export file (*.mzp) in the default export/import folder for your existing MashZone Server.

2. Copy the MZP export file you created in the previous step to the *MashZoneNG-install* /mashzone/data/importexport folder.
3. Click Admin Console in the MashZone NextGen Hub main menu.
4. Click **MashZone** to expand this section of the Administration menu.
5. Click **Import/Export**.
6. To import all MashZone feeds defined in any MZP export files in the *MashZoneNG-install* /mashzone/data/importexport folder, click **Import All** and confirm this choice.
   a. A validation status message is displayed. Click **OK**.

      The import begins. You become the owner for all the MashZone feed(s) that are imported.

   b. An import status message is displayed. Click **OK**.
7. To import specific MashZone feeds from the MZP export files in the *MashZoneNG-install* /mashzone/data/importexport folder, click **Import** and:
   a. If needed, change the **File Format** to see a list of specific export files.
   b. Select the MZP export file containing the MashZone feeds you want to import.
   c. Click **OK**.
   d. A validation status message is displayed. Click **OK**.

      The import begins. You become the owner for these MashZone feed(s).

   e. An import status message is displayed. Click **OK**.

You can now find the MashZone feeds in MashZone NextGen search results, open them, add views, create apps or add them to workspace apps. To allow other MashZone NextGen users to work with these feeds, you must also grant them permission to run these feeds. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.

Export MashZone Feeds from MashZone NextGen

You can export any or all MashZone feeds that you have imported into MashZone NextGen or created in MashZone NextGen to migrate them to another MashZone NextGen instance.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Import/Export** and select the **Export** tab.
4. To export specific MashZone feeds:
   a. Find and select the specific MashZone feeds you want to export from the list.
   b. Click **Export Selection**.

      This creates an MZP export file in the *MashZoneNG-install* /mashzone/data/importexport folder with a name in the form F*_feed-name _date-and-time* .mzp that you can use to migrate these MashZone feeds.
5. To export all MashZone feeds:
   a. Click **Export All**.
   b. When prompted, click **Export All** to confirm this export.

      This creates an MZP export file in the *MashZoneNG-install* /mashzone/data/importexport folder with a name in the form F*_first-feed-name _date-and-time* .mzp that you can use to migrate these MashZone feeds.

Enable the MashZone Feed Editor

By default, users cannot create new MashZone feeds in the Feed Editor. This editor is hidden from users in the MashZone NextGen Hub main menu.

You can make the Feed Editor visible to users in the **Mashups** menu in the MashZone NextGen Hub main menu to allow users to create new MashZone feeds.

Procedure

1. Open the file presto.conf in your text editor.

   MashZoneNG-install \apache-tomee-jaxrs\webapps\presto\WEB-INF\classes\presto.conf
2. Configure the `mashzone.feededitor.disabled` parameter.
   - Set the parameter `=false` to enable the MashZone Feed Editor.

     ```
     mashzone.feededitor.disabled=false
     ```
   - Set the parameter `=true` to disable the MashZone Feed Editor.

     ```
     mashzone.feededitor.disabled=true
     ```

3. Save your changes.
4. Reload the MashZone NextGen bowser tab.

This makes the **MashZone Feed Editor** in the MashZone NextGen Hub **Mashups** menu visible.

Enable the MashZone Administration

By default, users cannot use the MashZone Administration. This Administration is hidden from users in the MashZone NextGen Hub main menu.

You can make the MashZone Administration visible to users in the **Mashups** menu in the MashZone NextGen Hub **Admin Console** to allow users to manage specific MashZone features.

Procedure

1. Open the file presto.conf in your text editor.

   MashZoneNG-install \apache-tomee-jaxrs\webapps\presto\WEB-INF\classes\presto.conf

2. Configure the `mashzone.administration.disabled` parameter.
   - Set the parameter `=false` to enable the MashZone Feed Editor.

     ```
     mashzone.administration.disabled=false
     ```
   - Set the parameter `=true` to disable the MashZone Feed Editor.

     ```
     mashzone.administration.disabled=true
     ```
3. Save your changes.
4. Reload the MashZone NextGen bowser tab.

This makes the MashZone Administration in the MashZone NextGen Hub **Admin Console** visible.

To open the MashZone Administration click Admin Console in the MashZone NextGen Hub main menu. Then click **MashZone** to expand this section of the Administration menu.

Event Service Configuration and Administration

About the Event Service and Event Data

The Event Service that is integrated with MashZone NextGen allows MashZone NextGen to connect to the Event Bus for Software AG and subscribe to events published by other Software AG applications. In most cases, the system acting as the Event Bus is Universal Messaging or the webMethods Broker using a Messaging Service, although other systems and transports may serve as the Event Bus.

The Event Bus handles events published by *event producers* which may be a variety of Software AG applications. It routes events as they are published to *event consumers*, such as MashZone NextGen, who have subscribed to specific *event types*, also known as *channels*.

MashZone NextGen also uses the Event Service to connect to Apama and to subscribe to and work with events from *scenarios* (also sometimes called *dataviews*). Apama scenarios have event data that has been specifically transformed for use in dashboards.

For MashZone NextGen, each Event Bus or Apama subscription feeds an *event source* which is managed by the Event Service. Event sources receive events for subscriptions, store them in memory and act as the data source for the corresponding EDA or Apama event.

Use Events as Information Sources

To use events as information sources, you must create EDA or Apama event sources. Unlike other sources, only MashZone NextGen administrators can create EDA and Apama event sources for other users to work with.

To create these sources, administrators must:

1. For EDA event source "Identify the Event Type Store directory" on page 1706. This defines the types of events that are available for subscriptions from the EDA event source.
2. Create event sources of the Event Type Store and/or Apama as needed. MashZone NextGen retrieves event data from these event sources when users run the corresponding EDA or Apama event source. If views for the event source are real-time views, events are pushed to the view automatically.
   - Create EDA Event Sources
   - Create Apama Event Sources
3. "Start or Stop an Event Source" on page 1731

You can also Manage Apama Event Sources and Manage EDA Event Sources.

Manage EDA Event Sources

To identify, create, edit, delete, import or export *EDA Event Sources:*

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Select further steps:
   - "Identify the Event Type Store directory" on page 1706

- "Create EDA Event Sources" on page 1706
- "Edit EDA Event Sources" on page 1712
- "Duplicate EDA Event Sources" on page 1717
- "Delete EDA Event Sources" on page 1717
- "Import EDA Event Sources" on page 1718
- "Export EDA Event Sources" on page 1718

Identify the Event Type Store directory

The Event Type Store defines the different types of events that may be published by different applications through the Event Bus, and thus the types of events that users can subscribe to in MashZone NextGen. To begin configuration for event subscriptions and their associated event sources, you must first identify the Event Type Store directory.

To identiy the Event Type Store directory:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click **Event Type Store directory**.
6. Enter the path to the local Event Type Store in the **Event Type Store directory** field.
7. Click **Save**.

Your changes are applied.

Create EDA Event Sources

"You have identified the Event Type directory." on page 1706

Once you have identified the Event Type Store directory you can register subscriptions with the Event Bus. This creates *EDA Event Sources* that hold published events in memory and a corresponding event mashable in MashZone NextGen.

To create an EDA Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click **Create EDA Event Source**.

6. Set the properties for this event source. See table "EDA Event Source properties" on page 1707 below.
7. Click **Save**.

The EDA Event Source is created and listed by alias name.

Table 6. EDA Event Source properties

Property	Required	Description
Alias	yes	Enter a unique name for this event source.
Start event source automatically on server startup		This option is set by default, which automatically starts this event source when MashZone NextGen Server starts. Clear this option if you need to manually control startup for this event source.
EDA URL	yes	URL to Universal Messaging Service: nsp://<server>:<port> Example: nsp://localhost:9000]
Event type	yes	Whether events for this event source come from the Event Bus (an EDA connection) connection. Select the type of the event this event source should subscribe to. The XML schema files for these event types must exist in the Event Type Store directory.
Channel		Leave this property blank, unless the channel the event source should subscribe to is *not* the default channel for the selected event type. If the channel is not the default, select the correct channel.
Filter predicates		Enter a filter expression defining the events to be published to this event source. Detailed information on filtering events is available in documentation for Universal Messaging or the webMethods Broker, based on which system is acting as the Event Bus in your environment.
Check validity		Available only if **Strategy** is set to `Buffer`.

Property	Required	Description
		Determines whether saved events are valid with respect to the current time frame for the application (ta) and removes invalid events from the event source. ■ An event has a time stamp in the form of a time interval (I) = Start time - End time [ts - te); with ts being an element of I, and te not being an element of I. ■ The current time of the application is determined by the start time of the last received event. ■ An event is valid if the current time of the application is within the interval, i.e., [ts <= ta < te).
Preprocess and filter heartbeats		Removes empty events with no data from the event source. Empty events can, however, update the application time and thus can force a consolidation of the event source content.
Strategy	yes	The strategy that this event source uses for saving and removing events published from the Event Bus. Valid strategies are: ■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events. ■ `DELTA`= Events are stored by ID and added, updated or removed based on a command within the event. An event with an `Insert` command is saved in event source memory, any existing event with the same ID is overwritten. An event with a `Remove` command removes an existing event with the same ID.
Consider dimension		Available only when **Strategy** is set to `BUFFER`. Set this option to save events in separate series (or buckets) for each unique value of the **Dimension attribute**.
Dimension attribute	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set.

Property	Required	Description
		Select the event attribute whose unique values determine separate event series (buckets) for this event source.
Max. number of dimension values	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of unique dimension values (buckets) that can be tracked. Thus this is the maximum number of series that can store events. Default value: 1 (Max: 100.000). The product of **Max. number of dimension values** and **Capacity per dimension value** must not be greater than 100.000.
Dimension Squeeze-out		Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached. This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached. Set this option to change the bucket strategy to FIFO (first-in, first-out) which discards events for older series (buckets) and stores the newer event in a new series (bucket). Default value: false
Capacity per dimension value	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value. Default value: 10 The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000.

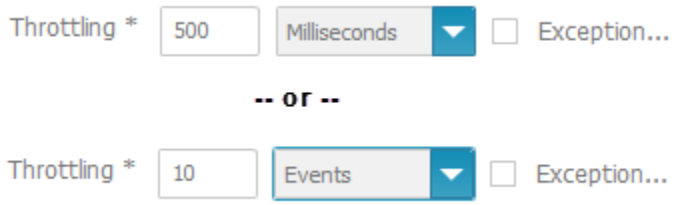
Property	Required	Description
Event ID attribute	yes	Available only when **Strategy** is set to `DELTA`. Select the attribute that identifies an event. The event ID and command determines which events are stored, updated or removed in this event source.
Command attribute	yes	Available only when **Strategy** if set to `DELTA`. Select the attribute that contains the event command (`Insert` or `Remove`). The event ID and command determines which events are stored, updated or removed in this event source.
Capacity	yes	Enter the maximum number of events to store in this event source. (Max: 100.000) Default value: 10
Memory model		Determines where events are stored: ■ `Internal`: the default which stores events in local memory for this event source. ■ `BigMemory`: stores events in a local BigMemory cache.
Throttling		Controls the speed and volume of event data that is pushed to views that subscribe to this event source. By default, event sources push event data every 500 milliseconds. You can: ■ Change the number of milliseconds to control throttling. ■ Change the measurement (Default=500) to `Events` to have throttling wait until a specific number of events are received and change the number, if needed. See "example below" on page 1711.
Exception		Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1): ■ A number

Property	Required	Description
		■ `Milliseconds` or `Events` as the measurement for the exception criteria See "example below" on page 1711

Simple and Hybrid Throttling Strategies

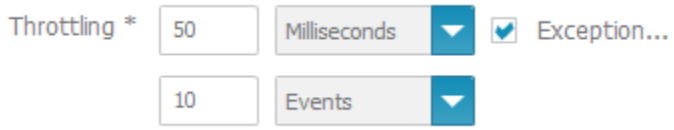
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events causes rendering issues.

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:



Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:



With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.
- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.

On the EDA Event Source overview page you can click on the **Alias** to show a preview of the specific Event Source properties.

Edit EDA Event Sources

You can edit already existing EDA Event Source.

Note: Changes in EDA connection properties can immediately affect data feed calculations so that they may not execute properly.

To edite an EDA Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the **Edit** icon to configure a specific EDA connection.
6. Set the properties for this event source:

Table 7. EDA Event Source properties

Property	Required	Description
Alias	yes	Enter a unique name for this event source.
Start event source automatically on server startup		This option is set by default, which automatically starts this event source when MashZone NextGen Server starts. Clear this option if you need to manually control startup for this event source.
EDA URL	yes	URL to Universal Messaging Service: nsp://<server>:<port> Example: nsp://localhost:9000]
Event type	yes	Whether events for this event source come from the Event Bus (an EDA connection) connection. Select the type of the event this event source should subscribe to. The XML schema files for these event types must exist in the Event Type Store directory.

Property	Required	Description
Channel		Leave this property blank, unless the channel the event source should subscribe to is *not* the default channel for the selected event type. If the channel is not the default, select the correct channel.
Filter predicates		Enter a filter expression defining the events to be published to this event source. Detailed information on filtering events is available in documentation for Universal Messaging or the webMethods Broker, based on which system is acting as the Event Bus in your environment.
Check validity		Available only if **Strategy** is set to `Buffer`. Determines whether saved events are valid with respect to the current time frame for the application (ta) and removes invalid events from the event source. ■ An event has a time stamp in the form of a time interval (I) = Start time - End time [ts - te); with ts being an element of I, and te not being an element of I. ■ The current time of the application is determined by the start time of the last received event. ■ An event is valid if the current time of the application is within the interval, i.e., [ts <= ta < te).
Preprocess and filter heartbeats		Removes empty events with no data from the event source. Empty events can, however, update the application time and thus can force a consolidation of the event source content.
Strategy	yes	The strategy that this event source uses for saving and removing events published from the Event Bus. Valid strategies are: ■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events. ■ `DELTA`= Events are stored by ID and added, updated or removed based on a command within the event. An event with an `Insert` command is saved in event source memory, any existing event

Property	Required	Description
		with the same ID is overwritten. An event with a `Remove` command removes an existing event with the same ID.
Consider dimension		Available only when **Strategy** is set to `BUFFER`. Set this option to save events in separate series (or buckets) for each unique value of the **Dimension attribute**.
Dimension attribute	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Select the event attribute whose unique values determine separate event series (buckets) for this event source.
Max. number of dimension values	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of unique dimension values (buckets) that can be tracked. Thus this is the maximum number of series that can store events. Default value: 1 (Max: 100.000). The product of **Max. number of dimension values** and **Capacity per dimension value** must not be greater than 100.000.
Dimension Squeeze-out		Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached. This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached. Set this option to change the bucket strategy to FIFO (first-in, first-out) which discards events for older series (buckets) and stores the newer event in a new series (bucket). Default value: false

Property	Required	Description
Capacity per dimension value	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value. Default value: 10 The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000.
Event ID attribute	yes	Available only when **Strategy** is set to `DELTA`. Select the attribute that identifies an event. The event ID and command determines which events are stored, updated or removed in this event source.
Command attribute	yes	Available only when **Strategy** if set to `DELTA`. Select the attribute that contains the event command (`Insert` or `Remove`). The event ID and command determines which events are stored, updated or removed in this event source.
Capacity	yes	Enter the maximum number of events to store in this event source. (Max: 100.000) Default value: 10
Memory model		Determines where events are stored: ■ `Internal`: the default which stores events in local memory for this event source. ■ `BigMemory`: stores events in a local BigMemory cache.
Throttling		Controls the speed and volume of event data that is pushed to views that subscribe to this event source. By default, event sources push event data every 500 milliseconds. You can: ■ Change the number of milliseconds to control throttling. ■ Change the measurement (Default=500) to `Events` to have throttling wait until a specific number of

Property	Required	Description
		events are received and change the number, if needed. See "example below" on page 1711.
Exception		Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1): ■ A number ■ `Milliseconds` or `Events` as the measurement for the exception criteria See "example below" on page 1711

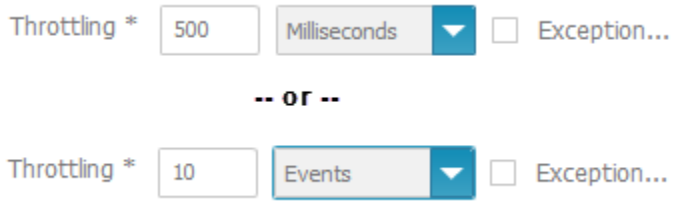
7. Click **Save**.

Your changes are applied.

Simple and Hybrid Throttling Strategies

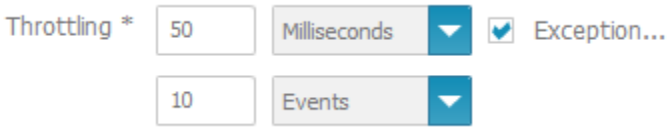
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events causes rendering issues.

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:



Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:



With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.
- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.
- see

Create EDA Event Sources

Duplicate EDA Event Sources

You can duplicate existing EDA Event Source.

To duplicate an EDA Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the **Copy** icon to duplicate a specific EDA Event Source.

The selected EDA Event Source is duplicated and listed with the prefix **copy_** in the **Alias**.

Delete EDA Event Sources

You can delete existing EDA Event Source.

Note: Deleting EDA Event Sources may cause data feeds to fail.

To delete an EDA Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.

3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the **Delete** icon to delete a specific EDA Event Source.

The selected EDA Event Source is deleted from the list.

Import EDA Event Sources

You can import individual configurations of EDA Event Sources. You can import only EDA configurations that are stored in the importexport directory MashZoneNG-install \mashzone\data\importexport or in one of the subfolders.

The files are saved as archive files, *.mzp. If an EDA Event Source with the same name already exists it cannot be imported. The existing Event Source must be deleted first.

To import an EDA Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click **Import EDA Event Source**.
6. Select an EDA .mzp file from the list:.
7. Click **Import EDA Event Source**.

The selected EDA Event Source with all relevant settings is imported and displayed.

Export EDA Event Sources

You can export the configurations of EDA Event Sources. The exported EDA Event Sources are stored directly in the importexport directory MashZoneNG-install\mashzone\data\importexport. The EDA Event Sources are saved with the corresponding settings and shared as *.mzp archive file.

To export an EDA Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the Export button to export an EDA Event Source.
6. Click **Close**.

The selected EDA Event Source is saved in the importexport directory as a mzp archive file.

Manage Apama Event Sources

To create, edit, delete, import or export **Apama** Event Sources:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Select further steps:
   - "Create Apama Event Sources" on page 1719
   - "Edit Apama Event Sources" on page 1724
   - "Duplicate Apama Event Sources" on page 1729
   - "Delete Apama Event Sources" on page 1730
   - "Import Apama Event Sources" on page 1730
   - "Export Apama Event Sources" on page 1731

Create Apama Event Sources

MashZone NextGen can work with events published from Apama through the Event Bus. In many cases, however, the events and data you need are defined in Apama*scenarios* which are not accessible through the Event Bus.

To work with Apama scenario events, you must create an Apama Event Source to receive scenario events.

To create new Apama Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click **Create Apama Event Source**.
6. Set the properties for this event source. See table "Table 8" on page 1719 below.
7. Click **Save**.

The *Apama Event Source* is created and listed by alias name.

Table 8. Apama Event Source properties

Property	Required	Description
Alias	yes	Enter a unique name for this event source.
Start event source automatically on server startup		This option is set by default, which automatically starts this event source when MashZone NextGen Server starts. Clear this option if you need to manually control startup for this event source.
Apama URL	yes	URL to the running **Apama** system (local or remote)
Apama Scenario	yes	Click **Refresh** to update the list of **Apama** scenarios for the selected Apama Event Source. If the **Apama** URL is set to a valid **Apama** system, then it is possible to select a scenario this event source should subscribe to.
Strategy	yes	The strategy that this event source uses for saving and removing events published from the Event Bus. Valid strategies are: ■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events. ■ `DELTA`= Events are stored by ID and added, updated or removed based on a command within the event. An event with an `Insert` command is saved in event source memory, any existing event with the same ID is overwritten. An event with a `Remove` command removes an existing event with the same ID. ■ `PARTIAL_EVENTS` = Each event has a unique identifier defined by one or more key fields (see Key attributes). Events contain additional fields, but may not contain all fields possible for the event. Simply put, each event may contain partial data. The event source maintains a single row for each unique event key representing the current full status for that event. Events published by Apama scenarios update the fields in that event source row that are included in the event, leaving other existing data for that event source row intact.

Property	Required	Description
		Events that have a new unique key are saved as a new row until event source memory is full. Once the event source memory is full, new events are discarded.
Consider dimension		Available only when **Strategy** is set to `BUFFER`. Set this option to save events in separate series (or buckets) for each unique value of the **Dimension attribute**.
Dimension attribute	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Select the event attribute whose unique values determine separate event series (buckets) for this event source.
Max. number of dimension values	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of unique dimension values (buckets) that can be tracked. Thus this is the maximum number of series that can store events. Default value: 10 (Max: 100.000). The product of **Max. number of dimension values** and **Capacity per dimension value** must not be greater than 100.000.
Dimension Squeeze-out		Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached. This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached. Set this option to change the bucket strategy to FIFO (first-in, first-out) which discards events for older series (buckets) and stores the newer event in a new series (bucket).

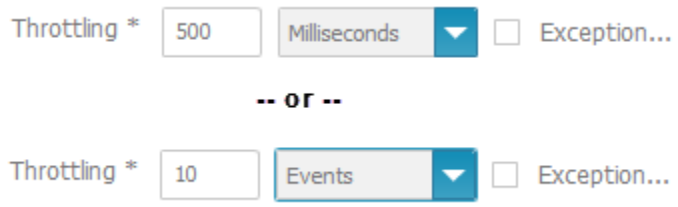
Property	Required	Description
		Default value: false
Capacity per dimension value	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value. Default value: 10; The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000.
Event ID attribute	yes	Available only when **Strategy** is set to `DELTA`. Select the attribute that identifies an event. The event ID and command determines which events are stored, updated or removed in this event source.
Command attribute	yes	Available only when **Strategy** if set to `DELTA`. Select the attribute that contains the event command (`Insert` or `Remove`). The event ID and command determines which events are stored, updated or removed in this event source.
Key attributes	yes	Available only when **Strategy** is set to `PARTIAL_EVENT`. The field(s) in events with partial data that uniquely identify an event. The event ID is used to ensure that events with partial data properly insert or update events in this event source. Select one or more attributes that uniquely identify events for this Apama scenario. If multiple fields are required, the order in which you select attributes determines how fields are combined to determine event IDs.
Capacity	yes	Enter the maximum number of events to store in this event source. (Max: 100.000) Default value: 10

Property	Required	Description
Memory model		Determines where events are stored: ■ `Internal`: the default which stores events in local memory for this event source. ■ `BigMemory`: stores events in a local BigMemory cache.
Throttling		Controls the speed and volume of event data that is pushed to views that subscribe to this event source. By default, event sources push event data every 500 milliseconds. You can: ■ Change the number of milliseconds to control throttling. ■ Change the measurement (Default=500) to `Events` to have throttling wait until a specific number of events are received and change the number, if needed. See "example below." on page 1723
Exception		Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1): ■ A number ■ `Milliseconds` or `Events` as the measurement for the exception criteria See "example below." on page 1723

Simple and Hybrid Throttling Strategies

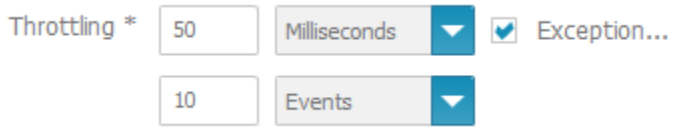
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events causes rendering issues.

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:



Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:



With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.
- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.

On the Apama Event Source overview page you can click on the **Alias** to show a preview of the specific Event Source properties.

Edit Apama Event Sources

You can edit an already existing Apama Event Source.

To edite an Apama Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the **Edit** icon to configure an Apama Event Source.

6. Set the properties for this event source. See table "Apama Event Source properties" on page 1725 below.
7. Click **Save**.

The *Apama Event Source* is created and listed by alias name.

Table 9. Apama Event Source properties

Property	Required	Description
Alias	yes	Enter a unique name for this event source.
Start event source automatically on server startup		This option is set by default, which automatically starts this event source when MashZone NextGen Server starts. Clear this option if you need to manually control startup for this event source.
Apama URL	yes	URL to the running **Apama** system (local or remote)
Apama Scenario	yes	Click **Refresh** to update the list of **Apama** scenarios for the selected Apama Event Source. If the **Apama** URL is set to a valid **Apama** system, then it is possible to select a scenario this event source should subscribe to.
Strategy	yes	The strategy that this event source uses for saving and removing events published from the Event Bus. Valid strategies are: ■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events. ■ `DELTA`= Events are stored by ID and added, updated or removed based on a command within the event. An event with an `Insert` command is saved in event source memory, any existing event with the same ID is overwritten. An event with a `Remove` command removes an existing event with the same ID. ■ `PARTIAL_EVENTS` = Each event has a unique identifier defined by one or more key fields (see Key attributes). Events contain additional fields, but may not contain all fields possible for the event. Simply put, each event may contain partial data.

Property	Required	Description
		The event source maintains a single row for each unique event key representing the current full status for that event. Events published by Apama scenarios update the fields in that event source row that are included in the event, leaving other existing data for that event source row intact. Events that have a new unique key are saved as a new row until event source memory is full. Once the event source memory is full, new events are discarded.
Consider dimension		Available only when **Strategy** is set to `BUFFER`. Set this option to save events in separate series (or buckets) for each unique value of the **Dimension attribute**.
Dimension attribute	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Select the event attribute whose unique values determine separate event series (buckets) for this event source.
Max. number of dimension values	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of unique dimension values (buckets) that can be tracked. Thus this is the maximum number of series that can store events. Default value: 10 (Max: 100.000). The product of **Max. number of dimension values** and **Capacity per dimension value** must not be greater than 100.000.
Dimension Squeeze-out		Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached.

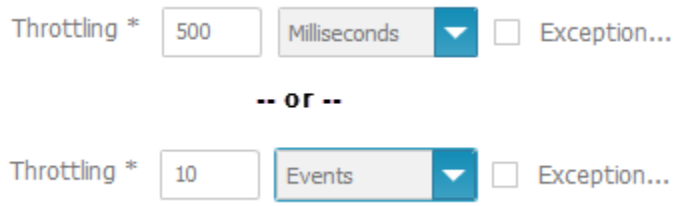
Property	Required	Description
		This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached. Set this option to change the bucket strategy to FIFO (first-in, first-out) which discards events for older series (buckets) and stores the newer event in a new series (bucket). Default value: false
Capacity per dimension value	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value. Default value: 10; The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000.
Event ID attribute	yes	Available only when **Strategy** is set to `DELTA`. Select the attribute that identifies an event. The event ID and command determines which events are stored, updated or removed in this event source.
Command attribute	yes	Available only when **Strategy** if set to `DELTA`. Select the attribute that contains the event command (`Insert` or `Remove`). The event ID and command determines which events are stored, updated or removed in this event source.
Key attributes	yes	Available only when **Strategy** is set to `PARTIAL_EVENT`. The field(s) in events with partial data that uniquely identify an event. The event ID is used to ensure that events with partial data properly insert or update events in this event source. Select one or more attributes that uniquely identify events for this Apama scenario. If multiple fields are required, the order in which you select attributes

Property	Required	Description
		determines how fields are combined to determine event IDs.
Capacity	yes	Enter the maximum number of events to store in this event source. (Max: 100.000) Default value: 10
Memory model		Determines where events are stored: ■ `Internal`: the default which stores events in local memory for this event source. ■ `BigMemory`: stores events in a local BigMemory cache.
Throttling		Controls the speed and volume of event data that is pushed to views that subscribe to this event source. By default, event sources push event data every 500 milliseconds. You can: ■ Change the number of milliseconds to control throttling. ■ Change the measurement (Default=500) to `Events` to have throttling wait until a specific number of events are received and change the number, if needed. See "example below." on page 1723
Exception		Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1): ■ A number ■ `Milliseconds` or `Events` as the measurement for the exception criteria See "example below." on page 1723

Simple and Hybrid Throttling Strategies

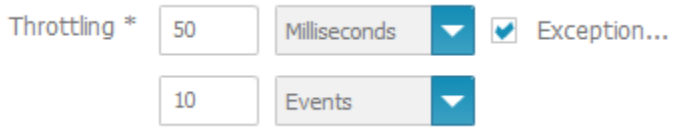
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events causes rendering issues.

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:



Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:



With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.
- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.

On the Apama Event Source overview page you can click on the **Alias** to show a preview of the specific Event Source propereties.

Duplicate Apama Event Sources

You can duplicate an existing Apama Event Source.

To duplicate an Apama Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.

4. Open the **Apama** tab.
5. Click the **Copy** icon to duplicate a specific Apama Event Source.

The selected Apama Event Source is duplicated and listed with the prefix **copy_** in the **Alias**.

Delete Apama Event Sources

You can delete an Apama Event Source.

Note: Deleting an Apama Event Source may cause data feeds to fail.

To delete an Apama Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the **Delete** icon to delete a specific Apama Event Source.
6. Click **Save**.

The selected Apama Event Source is deleted from the list.

Import Apama Event Sources

You can import individual configurations of Apama Event Sources. You can import only Apama Event Sources that are stored in the importexport directory MashZoneNG-install \mashzone\data\importexport or in one of the subfolders.

The files are saved as archive files, *.mzp. If an Apama Event Source with the same name already exists it cannot be imported. The existing Event Source must be deleted first.

To import an Apama Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click **Import Apama Event Source**.
6. Select an **Apama** .mzp file from the list:.
7. Click **Import Apama Event Source**.

The *Apama Event Source* with all relevant settings is imported and displayed.

Export Apama Event Sources

You can export the configurations of Apama Event Sources. The exported Apama Event Sources are stored directly in the importexport directory MashZoneNG-install \mashzone\data\importexport. The Apama Event Sources are saved with the corresponding settings and shared as *.mzp archive file.

To export an Apama Event Source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the Export button to export an **Apama** Event Source.
6. Click **Close**.

The selected **Apama** Event Source is saved in the importexport directory as a mzp archive file.

Start or Stop an Event Source

To begin receiving events from the Event Bus, you must start the event source configured for that event type. You can also stop individual event sources.

Note: Stopping an event source causes any existing events currently stored in memory to be deleted.

To stop/start an event source:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the event source tab **EDA** or **Apama** and either:
   - Select a specific event source and click **Start** to start just that event source.
   - Or select a specific event source and click **Stop** to stop that event source.

The selected event sources are stopped respectively started.

Restart all Event Source

To begin receiving events from the Event Bus, you must start the event source configured for that event type. You can restart all event sources at once.

Note: Restarting all event sources causes any existing events currently stored in memory to be deleted.

To restart all event sources:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Click **Restart all** to restart all event source instances of the selected event source type.

All event sources are restarted.

Process Performance Manager (PPM) Integration

Software AG Process Performance Manager (PPM) lets you discover and analyze processes that are not formally managed by a business process management solution (*BPMS*), such as webMethods BPMS. Using data sources throughout your enterprise, such as transactional data from your business systems, event streams from webMethods BPMS or database records from trading partners, PPM can model a process and assess its performance across various dimensions, such as region, product line, volume, or time. You can also use PPM's analytic tools to mine other data in your enterprise for meaningful patterns, trends, or correlations.

Information from PPM can be used in two ways in MashZone NextGen:

- As a source of data for a MashZone feed.
- As charts to add to a workspace app in Mashboard using the PPM Chart app. See "Add a PPM Chart to a Workspace" on page 1147 for more information.

Note: MashZone NextGen is compatible with PPM version 5.1.0 or above.

Connections to PPM are used in both cases. The tasks to integrate MashZone NextGen with PPM are:

1. Create PPM Connections for use in data sources or charts.

   You can also Manage PPM Connections.
2. Import the PPM Chart App to allow users to select and add PPM charts to workspace apps.

Manage PPM Connections

To edit, delete, import and export PPM connections

1. Click Admin Console in the MashZone NextGen Hub main menu.

2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**.
4. Follow the procedure of the remaining steps:
   - "Edit PPM Connections" on page 1734
   - "Delete PPM Connections" on page 1735
   - "Export PPM Connections" on page 1736
   - "Import PPM Connections" on page 1735

Create PPM Connections

You define connections for one or more PPM clients to allow users to use PPM as a data source for MashZone feeds or to allow users to add charts from PPM to workspace apps in MashZone NextGen.

Note: MashZone NextGen is compatible with PPM 5.1.0 or above.

For MashZone NextGen to connect and retrieve PPM data or charts, the following PPM applications must be started:

- PPM
- PPMClient

For details on your PPM installation, contact the system administrator in charge.You can enter PPM connection information manually or you can have MashZone NextGen determine them using the URL of a PPM favorite (favorites path). For information on copying the URL of a PPM favorite, see PPM on-line help topics.

To create PPM connections

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Connections** to expand this section of the Administration menu.
3. Click **PPM connections**.
4. Click **Create**.
5. Enter a name for the **PPM connections** in the **Alias** field, for example, the client name. The connection data is saved under this alias. Users may choose **PPM connections** by their alias.
6. To retrieve the connection data from the URL of a favorite from PPM:
   a. Click **Retrieve data**.
   a. Enter the URL of the PPM favorite that you copied earlier in the **URL** field.

a. Click **Resolve URL** to retrieve the required parameters from the URL. MashZone NextGen uses the favorite URL to complete the remaining fields for this connection.

7. To enter connection information manually:
   a. Select the protocol (HTTP or HTTPS) to use for the web application server that hosts the PPM query interface.
   a. In the **Host** field, enter the domain name or IP address of the PPM load balancer.
   a. In the **Port** field, enter the port number of the PPM load balancer.
   a. Specify the remaining context of the PPM query interface in the **Context** field. The context has the format **ppm/API_client name**, such as, API_umg_en.
8. Click **Check availability** to verify that the data is correct and that the PPM client is available.
9. Click **Save**.

The PPM connection is created and listed by alias name. This also lists the PPM version and availability of the PPM client.

Edit PPM Connections

You can edit already existing PPM connections.

Note: Changes in PPM connection properties can immediately affect data feed calculations so that they may not execute properly.

For MashZone NextGen to connect and retrieve PPM data or charts, the following PPM applications must be started:

- PPM
- PPMClient

For details on your PPM installation, contact the system administrator in charge.You can enter PPM connection information manually or you can have MashZone NextGen determine them using the URL of a PPM favorite (favorites path). For information on copying the URL of a PPM favorite, see PPM on-line help topics.

To edit PPM connections

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click the **Edit** icon to configure a PPM connection.
5. The **Alias** field of an already configured **PPM connection** is not editable. The connection data is saved under this alias.

6. To retrieve the connection data from the URL of a favorite from PPM:
   a. Click **Retrieve data**.
   a. Enter the URL of the PPM favorite that you copied earlier in the **URL** field.
   a. Click **Resolve URL** to retrieve the required parameters from the URL. MashZone NextGen uses the favorite URL to complete the remaining fields for this connection.
7. To enter connection information manually:
   a. Select the protocol (HTTP or HTTPS) to use for the web application server that hosts the PPM query interface.
   a. In the **Host** field, enter the domain name or IP address of the PPM load balancer.
   a. In the **Port** field, enter the port number of the PPM load balancer.
   a. Specify the remaining context of the PPM query interface in the **Context** field. The context has the format **ppm/API_client name**, such as, API_umg_en.
8. Click **Check availability** to verify that the data is correct and that the PPM client is available.
9. Click **Save**.

Your changes are applied.

Delete PPM Connections

You can delete existing PPM connections.

Note: Deleting PPM connections may cause data feeds or PPM Chart app to fail.

To delete PPM connections:

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click the **Delete** icon to delete a PPM connection.
5. Confirm the deletion.

The selected PPM connections are deleted from the list.

Import PPM Connections

You can import individual PPM connections.

The imported PPM connections are stored in the MashZoneNG-install/mashzone/data/importexport folder with the corresponding settings and shares as MashZone *.mzp

archive files. The export archive files have names in the form *A_PPM_PPM connection name_time stamp of export.mzp*.

To import PPM connections

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click the **Import**.
5. Click **Find** and select a PPM connection file.

The selected PPM connection with all relevant settings is imported and displayed.

Export PPM Connections

You can export individual PPM connections.

The exported PPM connections are stored in the MashZoneNG-install/mashzone/data/importexport folder with the corresponding settings and shares as MashZone *.mzp archive files. The export archive files have names in the form *A_PPM_PPM connection name_time stamp of export.mzp.*

To export PPM connections

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click **Export** to export a PPM connection.
5. Click **OK**.

The selected PPM connection is saved in the importexport directory as an archive file.

Import the PPM Chart App

The PPM Chart app, allows users to add charts to a dashboard for business process performance that were created in PPM. Users add PPM charts to workspace apps (dashboards) in Mashboard.

This custom app is available as an import package that is included in the MashZone NextGen installation. If integration with PPM is needed, you can simply import this custom app to make it available to users.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

2. Copy the *MashZoneNG-install* /mashzone/clientapps/ppm-chart.zip file to the *MashZoneNG-install* /prestocli/bin folder.
3. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
4. Enter this command:

   `padmin importApps -f ppm-chart.zip -s -u username-w password`

   With the appropriate account credentials. This typically in an administrator account.

   The `-s` flag ensures that this also imports the mashable or mashup dependencies for this custom app. There are other flags you may use. You may also specify a remote MashZone NextGen Server. See "Importing App Metadata" on page 1765 for more information and examples on this command.

   Messages and errors from the import process are sent to the console window (stdout).
5. Once the import is successfully finished, find and open this app in MashZone NextGen Hub. See "Find and Open Mashables, Mashups and Apps in MashZone NextGen Hub" on page 84 for instructions.
6. To allow other users to work with this app, add run permissions for appropriate users and groups. See "Grant Permission to Run Mashables, Mashups and Apps" on page 102 for instructions.

MashZone NextGen and MashZone Repositories

The MashZone NextGen Repository is the database that the MashZone NextGen Server uses to store meta-data, attributes and configuration for MashZone NextGen including:

- Artifacts (mashable information sources, mashups, and apps)
- Macros
- Taxonomies such as categories, tags and providers
- MashZone NextGen attributes (global, user and custom for artifacts)
- Configuration properties for the MashZone NextGen Server
- Snapshots taken of mashable or mashup results.

If you are using the default User Repository, user and group data is also stored in the MashZone NextGen Repository.

Important: TheMashZone NextGenRepository and MashZone Repository are initially installed in a Derby database suitable *only for trial*purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

Similarly, the MashZone Repository holds meta-data for the MashZone feeds and connections supported by the Integrated MashZone Server for MashZone NextGen.

Configuration and administration tasks for these two repositories include:

- Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution
- Support International Character Sets and Locales
- Use the Default MashZone NextGen User Repository
- Change MashZone NextGen Repository Ports
- "Tuning the MashZone NextGen or MashZone Repository Connection Pool" on page 1738
- Synchronize the MashZone NextGen Repository and MashZone NextGen Server Time Zones
- Sharing the MashZone NextGen Repository in Clustered Environments
- Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores
- Maintenance Suggestions

Maintenance Suggestions

Your existing standards for database backups, security and maintenance can be applied to the MashZone NextGen and MashZone Repositories. In addition, you should set up procedures to monitor or regularly manage growth for the MashZone NextGen Auditable Events table. This table tracks audit information for updates to the MashZone NextGen Repository.

You may also want to move snapshot data to a separate database to more easily manage growth and other operations for these datasets.

Tuning the MashZone NextGen or MashZone Repository Connection Pool

In addition to basic connection configuration, you can configure the connection pools for the MashZone NextGen Repository or the MashZone Repository. In many cases, you need to tune this configuration to optimize your MashZone NextGen environments.

Note: For a complete list of connection properties, see TomEE Datasource Properties.

To tune the connection pool, you update properties in the `<Resource>` element for the MashZone NextGen or MashZone repository in the *MashZoneNG-install* /apache-tomee-jaxrs/conf/tomee.xml file and then restart MashZone NextGen to apply these changes.

Connection Pool Size Properties

initialSize	The initial number of connections to create when the pool starts up. This defaults to 0.
maxActive	The maximum number of connections that can be allocated at one time. This defaults to 20. Set this to -1 to remove all limits.
maxWaitTime	The maximum number of milliseconds that the pool will wait when no connections are available before failing. Defaults to -1 which is an indefinite wait.

Idle Pool Connection Properties

maxIdle	The maximum number of connections that can be idle without connections being released. Defaults to 20. Set this to -1 to prevent any connections being released.
minIdle	The minimum number of idle connections that can exist before new connections are added to the pool. This defaults to 0, indicating no new connections should be created.
testWhileIdle	Whether connections should be tested when idle. If this is enabled, idle connections are tested using the `Validation query`. See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for more information on validation queries.
timeBetweenEvictionRuns	The number of milliseconds between tests of idle connections. This defaults to -1, which prevents all idle connection testing.
numTestsPerEvictionRun	The number of connections to test during any idle connection test run.
minEvictableIdleTime	The minimum number of milliseconds that a connection can be idle before being tested for eviction. Default is 3 minutes.

Synchronize the MashZone NextGen Repository and MashZone NextGen Server Time Zones

Creation and modification timestamps for artifacts and other MashZone NextGen Repository metadata can be different than times when events occurred in the MashZone NextGen Server in two cases:

- If the server hosting the MashZone NextGen Repository is located in a different time zone from the server hosting the MashZone NextGen Server
- If the time zone setting for the database hosting the MashZone NextGen Repository is set to a different time zone from the server hosting the MashZone NextGen Server

You can correct this problem by specifying a time zone in configuration for the MashZone NextGen Repository.

Important: The instructions in this topic are specific to MySQL databases. For other types of databases, please consult documentation for that database to determine the appropriate updates.

To synchronize time zones

1. In a text editor of your choice, open the rdsJDBC.properties file in your *MashZoneNG-config* folder. This is either in a shared external configuration folder or in *app-server:port* /presto/WEB-INF/classes.
2. Update the following properties:
   - `jdbc.url=jbdc:mysql://`*`hostname/database-name`*`?useLegacyDatetimeCode=false`
   - `dbServer.timeZone=`*`time-zone-id-for-mysql-server`*

     Valid time zone identifiers include GMT time zones, common abbreviations such as EST or UTC, or time zone locations such as `America/Los_Angeles`.
3. Save these changes.
4. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

MashZone NextGen Server Administration

Basic administration tasks for the MashZone NextGen Server include:

- Start and Stop the MashZone NextGen Server

  See also Startup Considerations.
- View MashZone NextGen Logs and Purge the Audit Log

- Manage Files for MashZone NextGen Features or Artifacts
- Deploying MashZone NextGen Instances, Clusters or Artifacts
- Clustering MashZone NextGen Servers

You may also be asked to use the MashZone NextGen Platform API Console by Technical Support during investigations of issues you report or as part of upgrade and migration tasks.

View MashZone NextGen Logs

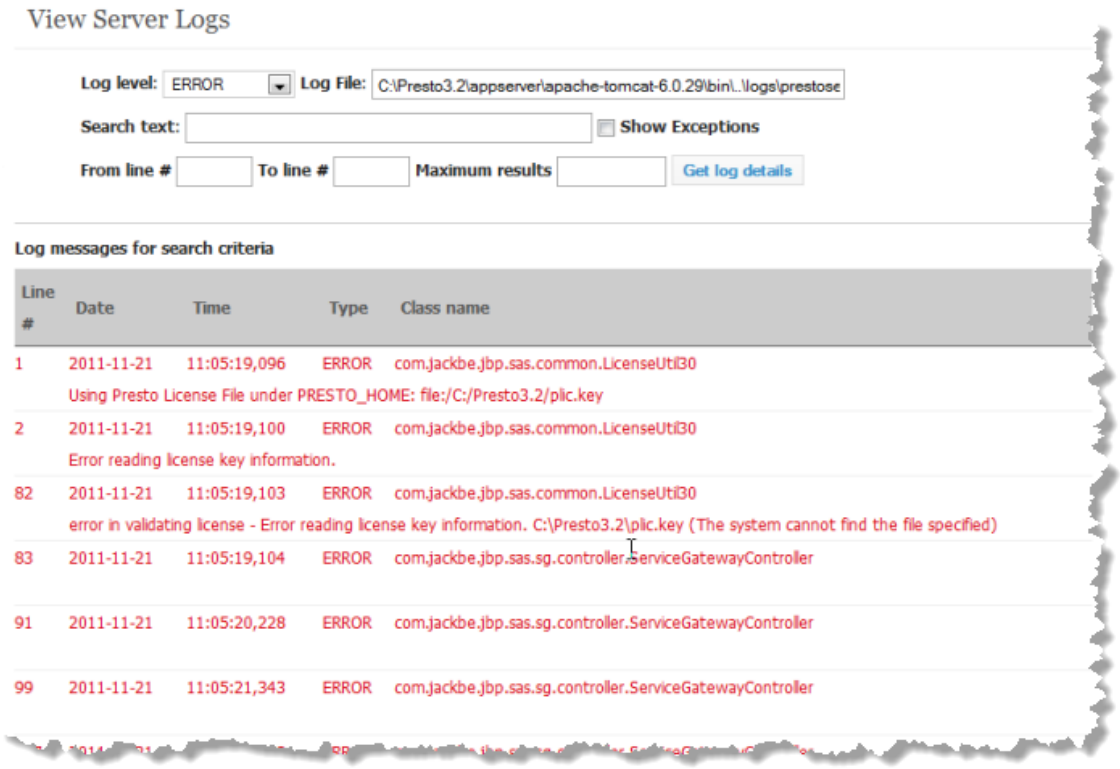
You can view two MashZone NextGen logs in MashZone NextGen Hub. See "View the MashZone NextGen Server Log" on page 1741 and "View the Audit Log for a Mashable, Mashup or App" on page 1742 for instructions. See also "MashZone NextGen Logging" on page 1650 for links to additional logging options for MashZone NextGen.

View the MashZone NextGen Server Log

To view the MashZone NextGen Server log, Click Admin Console in the MashZone NextGen Hub main menu. Then expand the **Audits and Logs** section and click **View Server Log**.

You can enter search criteria to limit the result to a specific logging level, matching text or specific lines within the log. You can also limit the number of results. Then click **Get log details**.

By default, this retrieves the most current prestoserver.log file and displays a one-line view of each log entry with a FATAL log level.



You can change the search criteria for:

- **Log level**: to see entries logged for any of the available logging levels.
- **Log file**: to see entries from earlier logs for the MashZone NextGen Server, update the file name to identify an earlier log file, such as prestoserver2.log.
- **Search text**: to search for log entries with a specific string. This search is case sensitive.
- **From line** and/or **To line**: to limit results to specific line numbers within the log file.
- **Maximum results**: to limit the number of log entries you want to see.

You can combine any of these search criteria. Set the **Show exceptions** option to see the full text for each log entry.

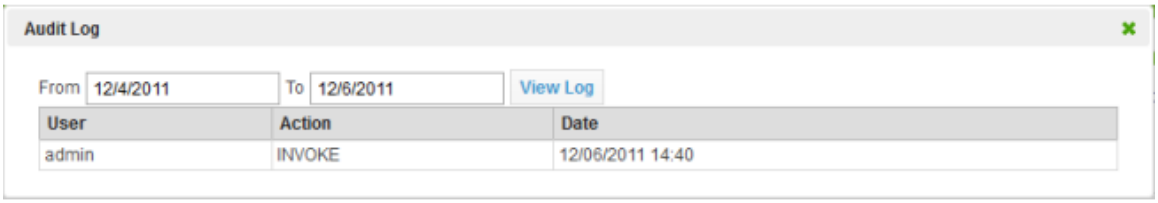
View the Audit Log for a Mashable, Mashup or App

The Audit Log can track each invocation or load for mashables, mashups or apps in MashZone NextGen as well as many other events for artifacts. This log is disabled by default.

If you have enabled the Audit Log and enabled logging for some artifact events, you can view log entries for a specific artifact:

1. Find the artifact in Search Results, favorites or other links and open this artifact.
2. Select **Show >** **Audit logs**.

3. If needed, update the start and end date to search for.
4. Click **View Log**.



Purge the Audit Log

The Audit Log tracks the invocation of each mashable or mashup. This logging is disabled by default (see "Turn Audit Logging On or Off" on page 1652).

Note: Purging the Audit Log permanently deletes audit data. You may wish to make a backup of this data in the MashZone NextGen Repository before completing this purge.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Audits and Logs** section and click **Audit Log**
3. To purge the entire Audit Log, click **Purge All**. To purge log entries for specific events, click **Purge** for that event.

Manage Pluggable Views and Libraries

You can use two screens in the Admin Console to manage the pluggable libraries and pluggable views that have been added to MashZone NextGen. See "Manage Pluggable Libraries" on page 1743, "Manage Pluggable Views" on page 1744 and "Manually Changing the Default Version for Libraries" on page 1745 for more information.

Manage Pluggable Libraries

Pluggable libraries and pluggable view libraries are added to MashZone NextGen and updated using import commands or Apache Ant build files. See "Creating Pluggable Views or Libraries" on page 1045 for instructions.

Note: Libraries of any kind frequently include several files which you can see in File Resources in the Admin Console. The library resources, however, should *always* be managed as a whole to ensure that MashZone NextGen configuration is up to date.

To manage pluggable libraries

1. Click Admin Console in the MashZone NextGen Hub main menu.

2. Expand the **Platform Features** section and click **Pluggable Libraries**.

   This lists information for the current default version of all pluggable libraries, including pluggable view libraries, added to the MashZone NextGen Repository.

3. To edit some properties for a pluggable library, click **Edit** for that library.

   Update properties as needed and click **Save**. See "Manually Changing the Default Version for Libraries" on page 1745 for more information on the effect of changes to the version property.

4. To delete *all* versions of a pluggable library, click **Delete** for that library.

   Note: Before you delete a pluggable library, be aware of any dependencies from other pluggable view libraries or custom apps.

Manage Pluggable Views

Pluggable view libraries are libraries that implement pluggable views shown in the MashZone NextGen View Gallery so that users may add this view to any number of mashables and mashups. They are added to MashZone NextGen and updated using import commands or Apache Ant build files. See "Creating Pluggable Views or Libraries" on page 1045 for more information.

Note: Libraries of any kind frequently include several files which you can see in File Resources in the Admin Console. The library resources, however, should *always* be managed as a whole to ensure that MashZone NextGen configuration is up to date.

To manage pluggable libraries

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Platform Features** section and click **Pluggable View Libraries**.

   This lists information for the current default version of all pluggable view libraries added to the MashZone NextGen Repository.

3. To edit some properties for a pluggable view library, click **Edit** for that library.

   Update properties as needed and click **Save**. See "Manually Changing the Default Version for Libraries" on page 1745 for more information on the effect of changes to the version property.

4. To delete *all* versions of a pluggable view library, click **Delete** for that library.

   Note: Before you delete a pluggable view library, be aware of any dependencies from views added to mashables or mashups or views used in basic or workspace apps.

Manually Changing the Default Version for Libraries

In most cases where you have multiple versions of a pluggable library or a pluggable view library in MashZone NextGen, you manage which version is the current default when you import a version of the library. The current default version determines which libraries are used with pluggable views and the basic or workspace apps that include these views. See "Managing Updates and Library Versions" on page 1085 for more information.

You can manually change which version of a library is marked as the default version, if needed, in both the Pluggable Libraries and View Libraries screens in the Admin Console.

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the **Platform Features** section and open **Pluggable Libraries** or **View Libraries**.
3. Find the pluggable library or pluggable view library you want to update and click **Edit** for that library.
4. Select the version you want to use as the default version in the **Version** property and click **Save**.

Manage Files for MashZone NextGen Features or Artifacts

MashZone NextGen uploads and hosts files that are used as schemas for Wires, as resources or help for custom apps or custom blocks in Wires, as thumbnails, as screenshots or all file associate with a pluggable library or pluggable view library. MashZone NextGen also uploads and hosts files for some types of mashables that are not accessible via HTTP (spreadsheets, CSV or XML files). These files are saved and managed in the MashZone NextGen Repository to ensure better management of resources and easier deployment or migration across different environments and versions.

Most files are added to the MashZone NextGen Repository automatically when users register, create or update the corresponding mashables, apps and macros. MashZone NextGen administrators may also need to manually add files to MashZone NextGen to provide common thumbnails, for resources used with custom Wires blocks or to register schemas for use in the Wires Mapper block.

Note: Mashups written in EMML may also use resources such as JavaScript files, Java classes, or other resources. With EMML, however, external resources are accessed through the classpath and cannot be uploaded and managed in MashZone NextGen.

Common management tasks for files include: Add External Resources as MashZone NextGen Files, Find MashZone NextGen Files or Update or Delete MashZone NextGen Files. See also "File Organization" on page 1747 for more information on file paths and URLs.

Note: Custom apps, pluggable libraries and pluggable view libraries typically have several resource files. In most cases, it is better to manage all the resources for custom apps and libraries as a whole to ensure that configuration for the apps and libraries is also correct. See "Manage Pluggable Views and Libraries" on page 1743and "App Packages and App Files" on page 1318 for more information.

Add External Resources as MashZone NextGen Files

You can add external resources to MashZone NextGen to make them easily accessible in custom blocks, custom apps, pluggable views, as thumbnails or many other purposes.

To add one or more files to MashZone NextGen

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Platform Features section and click **File Resources**.
3. Click **Upload New Files**.
4. Click **Browse**, find and select the file you want to upload and click **Open**.

   The location and file name fill in and a new set of fields open to upload another file.
5. If needed, add to the path or change the file name.

   The name of the file defaults to `/`*file-name* . If you accept the default, the URL to access this file becomes `http://`*app-server:port* `/presto/files/`*file-name* .

   You can organize files into 'pseudo folders' by adding to the path, using `/` as the separator. For example, a file name of `/images/reports.png` has a URL of `http://`*app-server:port* `/presto/files/images/reports.png` and can be found in file search (along with any other files in the 'images folder') by searching for `images` as the file name.

   You can also upload files that are normally loaded automatically, such as thumbnails. Simply specify the standard path. See "File Organization" on page 1747 for more information.
6. Repeat the steps, as needed, to find and name any other files you want to upload.
7. Click **Delete** to close the empty file upload fields.
8. Click **Upload files**.

   The files are added to the MashZone NextGen Repository and are now available via a MashZone NextGen URL.

Find MashZone NextGen Files

To find files that have been uploaded to MashZone NextGen

1. Click Admin Console in the MashZone NextGen Hub main menu.

2. Expand the Platform Features section and click **File Resources**.
3. Enter either:
   - Part of the file name(s).
   - Part of the path to the file(s). See "File Organization" on page 1747 for a list of the standard paths that MashZone NextGen uses.
4. Click **Search**.

> **Note:** File search results are always sorted by path and file name.

Update or Delete MashZone NextGen Files

Although rare, you may occasionally need to update or even delete files from MashZone NextGen.

To update or delete a file

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Expand the Platform Features section and click **File Resources**.
3. Find the specific file you need to update or delete. See "Find MashZone NextGen Files" on page 1746 for techniques.
4. To upload an updated file:
   a. Click **Edit** on the line for that file.
   b. Click **Browse** and find the updated file you want to replace the existing file in MashZone NextGen.
   c. Click **Upload this file**.
5. To delete a file, click **Delete** on the line for that file.

File Organization

File names are path-like to support both URL access and common file organization techniques. Files that are uploaded automatically, such as resources for custom apps or views, are organized in the `/system` 'folder.' Files that you upload manually default to the 'root folder' of `/`. You can, however, define any level of folder organization you need by defining your own folder paths when you manually upload files.

Standard file paths include:

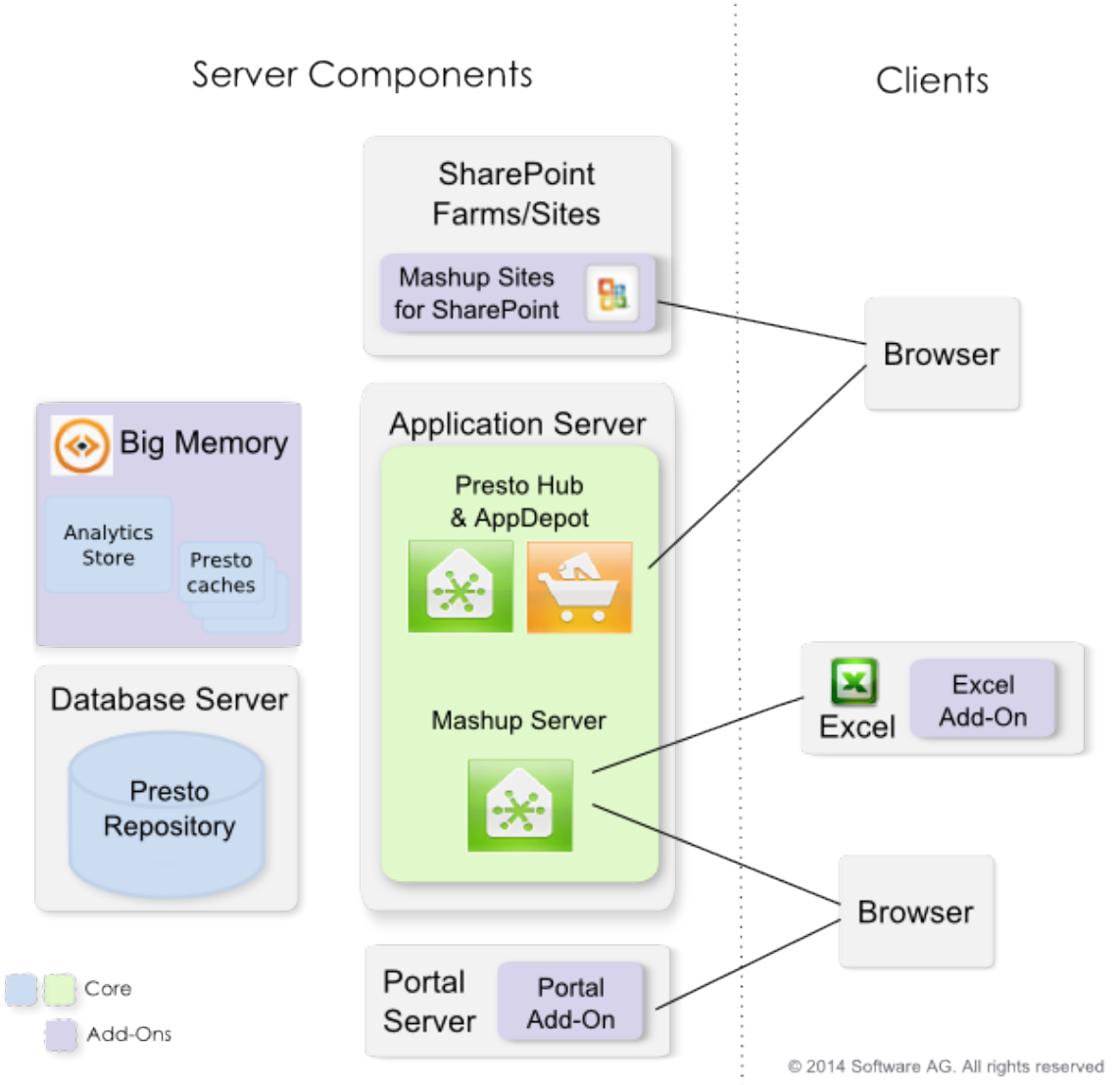
- `/system/lib`: the foot path for all pluggable views.
- `/system/mashlets`: the root path for all custom apps.
- `/system/mashlets/`*`app-id`*: the root path for all resource files for a specific custom app.

- `/system/mashlets/app-id/js`: JavaScript libraries for a specific custom app
- `/system/mashlets/app-id/css`: CSS stylesheets for a specific custom app.
- `/system/mashlets/app-id/screenshots`: screenshots for a specific custom app.
- `/system/thumbnails`: root folder for thumbnails.
- `/system/wires/schemas`: root folder for all registered XML schemas available for use in the Mapper block.

To determine the URL to access a file, add `http://app-server:port /presto/files` to the file name.

Deploying MashZone NextGen Instances, Clusters or Artifacts

Deploying MashZone NextGen to new hosts or new environments typically involves Deploying the Core Components, shown below, and optionally Deploying Server and Client Add-Ons or "Deploying MashZone NextGen Artifacts and Other Metadata" on page 1750.



Deploying the Core Components

The core components include the MashZone NextGen Server, MashZone NextGen Hub and AppDepot (*web-apps-home* /presto), the MashZone NextGen Repository, which is typically installed in a database other than the default Derby database, and the MashZone NextGen Analytics In-Memory Stores and MashZone NextGen caches.

Note: In earlier releases, the MashZone NextGen Hub and AppDepot were deployed in a separate web application from the MashZone NextGen web application. Effective in 3.2, all the core components are deployed in the single *web-apps-home* /presto web application.

For individual MashZone NextGen servers, you typically do a default installation (see *Installing Software AG Products*). You may also move the MashZone NextGen Repository

to a database of your choice. See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for instructions.

You can leave the the MashZone NextGen Analytics In-Memory Stores and MashZone NextGen caches in local memory for a single MashZone NextGen server. This uses the default client installation of BigMemory. If additional memory or reliability is required, you can also deploy BigMemory as an add-on in a separate host or cluster. See "Working with MashZone NextGen Analytics In-Memory Stores" on page 857 for more information and links.

To deploy multiple unclustered servers, see "Deploying Multiple MashZone NextGen Servers in One Host" on page 1771. To deploy MashZone NextGen servers in clusters, see "Clustering MashZone NextGen Servers" on page 1771 for requirements and links.

Deploying Server and Client Add-Ons

You may also deploy two server-side add-ons: MashZone NextGen Add-On for SharePoint and the MashZone NextGen Add-On for Portals. See "Installing MashZone NextGen Add-On for SharePoint" on page 1366 and "Installing the MashZone NextGen Add-On for Portals" on page 1342 for instructions.

Users may also need to install the MashZone NextGen Add-On for Excel, an add-on toolbar for Excel, to work with mashables and mashups in Microsoft Excel or to publish spreadsheets as mashables. See "Install the MashZone NextGen Toolbar in Excel" on page 1323 for details.

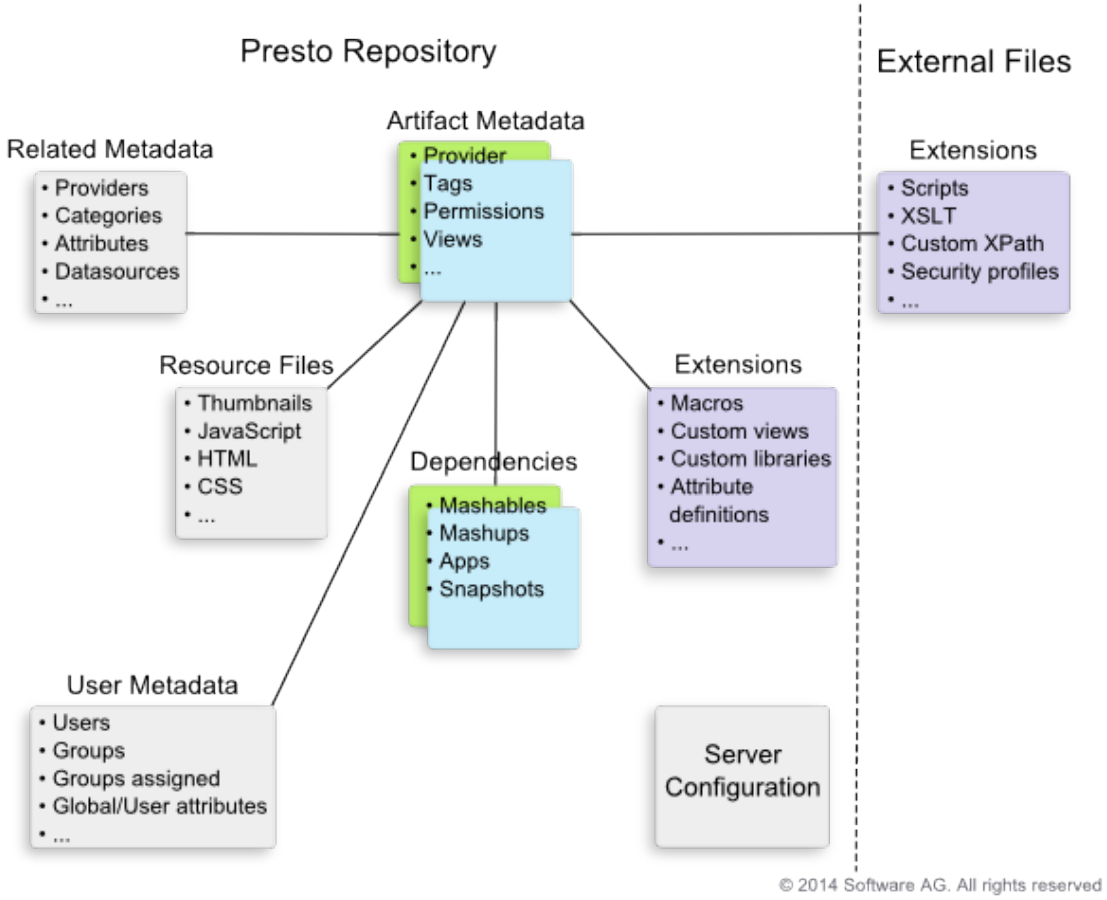
Deploying MashZone NextGen Artifacts and Other Metadata

You deploy specific artifacts (mashables, mashups or apps) and other metadata from a source MashZone NextGen Server to a target MashZone NextGen Server using the export and import commands.

Important: You *cannot* use export and import commands when the MashZone NextGen version for the source and target MashZone NextGen Servers are different:

- For major upgrades, use the migrate command instead.
- For minor upgrades, please contact Technical Support or your Software AG representative.

In addition to the basic metadata for an artifact, a successful deployment must include related metadata, related files, extensions the artifact may use and any other artifacts that the artifact depends on.



The export and import commands automate deployment for most of this data, with some specific limitations that require manual deployment steps.

1. Export the specific mashable, mashup or app artifacts that you want to deploy to another MashZone NextGen Server and any macros, pluggable views or pluggable libraries that they may use.

   See the following topics for instructions using these MashZone NextGen export commands:

   - "Exporting Mashable and Mashup MetaData" on page 1755
   - "Exporting Macros" on page 1756
   - "Exporting App MetaData" on page 1758
   - "Exporting Pluggable Views or Libraries" on page 1759
   - "Exporting MashZone NextGen Global Attributes" on page 1762
   - "Exporting Users, User Metadata and Groups" on page 1763

   The data that is exported and known limitations for these commands include:

Table 10. Known Export/Import Limitations

	Exported	Not Exported
Artifact Metadata	■ Basic metadata such as provider, category, tags and description. ■ Ownership (who created the artifact). ■ On/off status. ■ Run permissions. ■ Views. ■ Artifact attributes. ■ For apps, the AppDepot status.	■ User rating and feedback. ■ Snapshots. ■ Snapshot schedules. ■ Caching configuration.
Related Metatdata/ User Metadata	■ Providers. ■ Categories. ■ Global and user MashZone NextGen attributes. ■ Users, groups and user group assignments if this data is tracked in the default MashZone NextGen User Repository and not in your LDAP Directory. ■ Configuration for SharePoint connections used by SharePoint mashables or mashups.	■ Datasources and their JDBC drivers that are used by database mashables or by mashups. Datasources *must* be added to the target MashZone NextGen Server before you import any artifacts that use them or the import will fail. ■ For apps that are published to the AppDepot, any user preferences for Favorite Apps. ■ User preferences for apps in Mashboard or the Mashboard state for workspace apps.
Resource Files	■ Thumbnails for apps or pluggable views. ■ Screenshots for apps. ■ HTML, JavaScript, CSS, images or any other file uploaded in the package for a custom app.	Thumbnails for mashables or mashups.

	Exported	Not Exported
Dependencies	Optionally can export dependent artifacts: ■ For workspace apps, this always exports all the apps used in the workspace. ■ For individual apps, you can choose to also export any mashables or mashups *explicitly declared and used* by those apps. If you choose to include dependencies, all dependencies for basic apps are handled because MashZone NextGen automatically declares dependencies for basic apps. For custom apps, export handles any dependencies that are explicitly declared with a <dependson> element in the App Specification. App developers must supply this information. ■ For mashups, this always exports any other mashups or mashables that are used by the mashup. ■ For pluggable views or pluggable libraries you can choose to export any library dependencies.	Any snapshots used by apps.
Extensions	■ Registered macros, for use in mashups. ■ HTML, JavaScript, CSS, images or any other file uploaded in the package for a pluggable view or pluggable library.	■ Attribute definitions for extension attributes in artifacts. ■ Any of the file-based extensions such as custom XPath functions. See for a complete list.

Exported	Not Exported
MashZone NextGen Server Configuration	Configuration for the MashZone NextGen Server.

2. Copy the files for any extensions used by the exported artifacts from the *MashZoneNG-config* folder for the source MashZone NextGen Server to the *MashZoneNG-config* folder for the target MashZone NextGen Server. See for a list of potential file-based extensions.

   Note: The *MashZoneNG-config* folder may be an external configuration folder outside of the source and target MashZone NextGen Servers or it may be in the default locations. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information on *MashZoneNG-config* locations.

3. Define *datasources* in the Admin Console for the target MashZone NextGen Server with *matching* names and JDBC drivers to the datasources in the source MashZone NextGen Server. If these are not present, import of database mashables or mashups that use datasources fails.

   See "Manage Data Sources and Drivers" on page 1679 for instructions.

4. If the exported artifacts include SharePoint mashables or mashups that use SharePoint as an information source, make sure the MashZone NextGen license for the target MashZone NextGen Server includes the MashZone NextGen Add-On for SharePoint Add-On.

5. Use the export files created earlier to import mashables, mashups, apps, macros, pluggable views, pluggable libraries, MashZone NextGen global and user attributes, users, groups and user group assignments from the source MashZone NextGen Server.

   See the following topics for information on using these commands:

   - "Importing Macros" on page 1764
   - "Importing Mashable or Mashup MetaData" on page 1763
   - "Importing Pluggable Views or Libraries" on page 1767
   - "Importing App Metadata" on page 1765
   - "Importing MashZone NextGen Global Attributes" on page 1769
   - "Importing Users, User Metadata and Groups" on page 1770

6. If needed, Update Mashable Endpoints or Update Mashable Security Profiles.

Exporting Mashable and Mashup MetaData

Exporting mashables or mashups exports their metadata from one MashZone NextGen Repository to a file. You can then import this file to another MashZone NextGen Repository.

Typically, you export and import mashables and mashups to move new artifacts to production or to replicate data for a new instance of the MashZone NextGen Server.

Important: This command has specific limitations for what it exports. See "Table 10" on page 1751 for details.

1. If it is not running, start the MashZone NextGen Server for the MashZone NextGen Repository that is the source for the mashables or mashups that you wish to export. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin exportServices -q filter -f output-file
[-l prestoURL] -u username -w password
[-v]
```

   - *-q filter* : defines which mashables or mashups to export. It can be one of:
     - `"all"` = export all mashables and mashups from this MashZone NextGen Repository. You can also use `"ALL"` or `"*"`.
     - `"ids=`*list-of-artifact-ids*`"` is a comma-separated list, with no spaces, of the exact IDs of the mashables or mashups that you wish to export.
     - `"name=`*artifact-name-pattern or list-of-artifact-names*`"` is one specific artifact name, a portion of an artifact name with wildcards, such as `Yah*` or`*Yah`, or a comma-separated list of artifact names, with no spaces.
     - `"tag=`*some-tag*`"` is the name of a user-defined tag. This selects all mashables or mashups that have that tag. You can also use wildcards to select by partial tag name, such as `News*` or `*News`. Use `*` to export all mashables and mashups that have tags. Mashables and mashups with no tags are excluded.
     - `"type=`*artifact-type*`"` = ATOM, DATABASE, MASHUP, RSS, REST, SHAREPOINT, SPREADSHEET or WSDL. This selects all artifacts of that type.
   - *-f output-file* : is the path and name of the export file to hold the metadata.
   - *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
   - *-w password* : is the MashZone NextGen password to log in with.

- *-v*: is an optional flag to turn on verbose logging.

General messages and errors from the export process are sent to the command window (stdout). Messages for specific artifact failures are included in the export file in <FailedExport> elements. Once the export command completes successfully, you can use the output file to import mashables or mashups to another MashZone NextGen Repository.

Example

The following example from a Windows environment, exports data for all REST mashables from the localhost to a file named `localRestSvcs.xml` and logs all messages or errors from the export process to a file named `localRestExport.log`.

```
c:\MashZone NextGenversion\prestocli> padmin exportServices
  -q "type=REST" -f localRESTSvcs.xml -u Administrator -w manage
  >> localRestExport.log
```

Exporting Macros

Exporting macros exports their metadata from one MashZone NextGen Repository to a file. You can then import this file to another MashZone NextGen Repository.

Typically, you export and import macros along with the mashups that use them to move new mashups to production or to replicate data for a new instance of the MashZone NextGen Server. You can also use export and import for macros to make new custom blocks available in Wires for other instances of the MashZone NextGen Server.

1. If it is not running, start the MashZone NextGen Server for the MashZone NextGen Repository that is the source for the macros that you wish to export. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin exportEmmlMacro -f output-file
[-d domain -g -n macroName -l prestoURL]
-u username -w password
[-v]
```

   - *-f output-file* : is the path and name of the export file to hold the metadata.
   - *-d domain* : the macro domain containing the macro(s) to export. The *domain* value can be:
     - `all` or `ALL` = export all macros in all domains from this MashZone NextGen Repository. This omits global macros.
     - *domain-name* is the name of one specific domain that contains the macro(s) you want to export.

   Note: This option is mutually exclusive with the `-g` option.

   If neither `-d` or `-g` as specified, all macros are exported.

- *-g*: to export global macro(s). If no macro name is included with the `-n` option, this exports all global macros and omits macros in any custom domain.

  Note: This option is mutually exclusive with the `-d` option.

  If neither `-d` or `-g` as specified, all macros are exported.

- *-n macroName* : the name of the specific macro to export. You must also specify the domain for this macro with the `-d` option or use the `-g` option if this is a global macro.
- *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
- *-w password* : is the MashZone NextGen password to log in with.
- *-v*: is an optional flag to turn on verbose logging.

General messages and errors from the export process are sent to the command window (stdout). Messages for specific artifact failures are included in the export file in <FailedExport> elements. Once the export command completes successfully, you can use the output file to import macros to another MashZone NextGen Repository.

Examples

The combinations of the `-d`, `-g` and `-n` options give you precise control of the macros you want to export. This example exports all macros, both global and custom domains, from the MashZone NextGen Server in the local host to a file named allMacros.xml:

```
padmin exportEmmlMacro -f allMacros.xml -u Administrator -w manage
```

This next example export the macros from the MashZone NextGen Server at `presto12.myorg.com:8080` in the domain named `Finance`:

```
padmin exportEmmlMacro -f financeMacros.xml -d Finance -l presto12.myorg.com:8080 -u Administrato
```

This example exports all macros in custom domains from the MashZone NextGen Server in the local host:

```
padmin exportEmmlMacro -f domainMacros.xml -d ALL -u Administrator -w manage
```

While this example exports all global macros from the same MashZone NextGen Server:

```
padmin exportEmmlMacro -f globalMacros.xml -g -u Administrator -w manage
```

This final example exports the global macro named `computeBasicAuth`:

```
padmin exportEmmlMacro -f basicAuthMacro.xml -g -n computeBasicAuth -u Administrator -w manage
```

Exporting App MetaData

Exporting apps exports both metadata and *any* associated files including screen captures, thumbnails, HTML files, JavaScript files, CSS files, image files or any other file define in the App Specification for that app.

This includes basic and custom apps as well as apps that have been published to the AppDepot. Export also exports any apps used in workspace apps that are being exported and, optionally, can export other artifacts that an app depends on.

Export creates an export file that you can then use to import apps to another MashZone NextGen Repository. Typically, you export and import apps to move new apps to production or replicate data to a newly deployed MashZone NextGen Server.

Important: This command has specific limitations for what it exports. See "Deploying MashZone NextGen Artifacts and Other Metadata" on page 1750 for details.

1. If it is not running, start the MashZone NextGen Server for the MashZone NextGen Repository that is the source for the apps you wish to export. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin exportApps -q filter [-f output-file -l prestoURL]
-s -u username -w password [-v] [-o]
```

   - *-q filter* : defines which apps to export. The filter can be:
     - `"all"` to export all apps from this MashZone NextGen Repository. You can also use `"ALL"` or `"*"`.
     - `"author=`*list-of-owner-ids*`"`, the user ID for a specific user or a list of comma-separated user IDs. This exports apps created by those users.
     - `"category=`*list-of-categories*`"`, a specific category or a list of comma-separated categories. This exports apps with those categories. Apps with no category are not included.
     - `"ids=`*list-of-app-ids*`"`, a specific app ID or a list of comma-separated app IDs. This exports those specific apps.
     - `"mine"`, to export all apps that were created by the user identified by the credentials used to execute this command (in the `-u` option).
     - `"provider=`*list-of-providers*`"`, a specific provider or a list of comma-separated providers. This exports apps with those providers. Apps with no provider are not included.
     - `"tag=`*list-of-tags*`"` , a specific user-defined tag or a list of comma-separated tags. This exports apps with those tags. Apps with no tags are not included.

- *-f output-file* : an optional path and name for the export file to put app data into. If omitted, this generates an output file in the folder where this command is executed with a name of either:
  - app-export-*yymmdd-hhmm* .xml
  - app-export-*yymmdd-hhmm* .zip

  Important: Exporting apps on Linux systems can fail with an error that the output file cannot be created. To avoid this error, specify an output file in the tmp folder for your account, such as:

  ```
  -f /users/userA/tmp/my-apps-export.xml
  ```

  If you supply a path and file name, the export file is created with the name you specify, *except* for the file extension. If the apps you export do not have any screen captures, thumbnails or any other associated files (such as JavaScript), then the export file extension is XML.

  If any of the apps you export have screen captures, thumbnails or other associated files, then the export file has a ZIP extension. This archive contains both the XML export file and all of the associated files for those apps that have them.

  This file must not already exist, unless you also use the `-o` option.
- *-o*: an optional flag to allow export information to overwrite an existing export file. If you omit this option, the output file must not already exist.
- *-s*: an optional flag to also export any mashables or mashups that are used by an app that is being exported.
- *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
- *-w password* : is the MashZone NextGen password to log in with.
- *-v*: is an optional flag to turn on verbose logging.

Messages and errors from the export process are sent to the command window (stdout). Once the export command completes successfully, you can use the output file to import apps to another MashZone NextGen Repository.

Exporting Pluggable Views or Libraries

Exporting pluggable views or custom, named libraries exports both configuration information and the file resources for those views or libraries from the MashZone NextGen Repository. The output of an export may be either:

- A zipped archive suitable to use as the input to deploy those views or libraries to another MashZone NextGen Repository. See the "Example 153. Examples" on page 1761 section for more information.
- A directory with the fully expanded configuration and resource files suitable for editing to update the pluggable view or pluggable library. See for instructions and an example.

Important: If MashZone NextGen hosts several versions of pluggable views or libraries, *only the default version* of each view or library is exported.

Pluggable views and libraries typically consist of configuration information for MashZone NextGen plus one or more JavaScript, CSS or image files. Pluggable views may also include a thumbnail image that displays in the View Gallery. See "Views and Libraries in MashZone NextGen" on page 1045 for more information.

1. If it is not running, start the MashZone NextGen Server for the MashZone NextGen Repository that is the source for the pluggable views or shared libraries that you wish to export. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin exportLib -q filter [-s] [-f output-file]
[-d output-directory] [-o] [-l prestoURL]
-u username -w password [-v]
```

   - *-q filter* : defines which pluggable views or libraries to export. The filter can be:
     - `"all"` to export all pluggable views and pluggable libraries from this MashZone NextGen Repository. You can also use `"ALL"` or `"*"`.
     - `"author=`*list-of-owner-ids*`"`, the user ID for a specific user or a list of comma-separated user IDs. This exports pluggable views and pluggable libraries registered by those users.
     - `"mine"`, to export all pluggable views or libraries that were registered by the user identified by the credentials used to execute this command (in the `-u` option).
     - `"ids=`*list-of-library-ids*`"`, a specific ID for a pluggable view or library or a list of comma-separated IDs for views or libraries. This exports those specific views or libraries.
     - `"name=`*list-of-library-names*`"`, a specific pluggable view or library name. ?list also?
     - `"type=view"`, to export only pluggable views.
     - `"subtype=`*view-category-name*`"`, to export only pluggable views that belong to the specified view category.

- *-s*: an optional flag to also export any named libraries that are used by the pluggable views or libraries that are being exported.
- *-f output-file* : is the path and name of the export file to hold both configuration and resource files for the export. The export file is always a ZIP archive, with .zip extension, which is suitable to deploy the exported views and libraries to a different MashZone NextGen Repository.

  Note: This option is mutually exclusive with the `-d` option.

  If you omit both the `-f` and `-d` options, the export produces a ZIP file named app-export-*date*-*time* .zip in the directory where you execute this command.
- *-d output-directory* : is the path to the directory where export configuration plus resources for the exported pluggable views and libraries should be output. All resources are fully expanded, allowing you to update view and library resources. Use the `importLib` command to upload these updates to MashZone NextGen.

  Note: This option is mutually exclusive with the `-f` option.

  If you omit both the `-f` and `-d` options, the export produces a ZIP file named app-export-*date*-*time* .zip in the directory where you execute this command. This ZIP archive is suitable as the input for the `importLib` command to deploy the exported views and libraries to a different MashZone NextGen Repository.
- *-o*: an optional flag to allow export information to overwrite an existing export file or directory. If you omit this option, the output file must not already exist or the output directory must be empty.
- *-l prestoUrl* : an optional URL to the MashZone NextGen Server to export views and libraries from. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.

  Use this option if the MashZone NextGen Server is remote, if it is not running in Tomcat or if it not using the default Tomcat port.
- *-u username* : the MashZone NextGen username to use as credentials to execute this command. This account *must* have MashZone NextGen administrator permissions.
- *-w password* : the MashZone NextGen password to use as credentials to execute this command.
- *-v*: an optional flag to turn on verbose logging.

General messages and errors from the export process are sent to the command window (stdout). Messages for specific library failures are included in the export file in <FailedExport> elements.

Examples

The following example from a Windows environment, exports all pluggable views and libraries from the local MashZone NextGen Server to a file named

`localCustomViews.zip` that can then be used to deploy these libraries to another MashZone NextGen Server using the importLib command.

```
c:\MashZone NextGenversion\prestocli> padmin exportLib
  -q "all" -f localCustomLibs -u Administrator -w manage
```

This next example, exports only pluggable views from a remote MashZone NextGen Server to a file named `remoteCustomViews.zip` that can then be used to deploy these libraries to the local MashZone NextGen Server using the importLib command.

```
c:\MashZone NextGenversion\prestocli> padmin exportLib
  -q "type=view" -f remoteCustomViews
  -l http://204.87.1.110:8080/presto/edge/api -u Administrator -w manage
```

For other exportLib examples, see "Export Resources for an Existing Pluggable View or Library for Updates" on page 1089.

Exporting MashZone NextGen Global Attributes

You can export metadata for all global MashZone NextGen Attributes from the MashZone NextGen Repository to an export file. You can then import this file to another MashZone NextGen Repository.

Note: This command is available only in MashZone NextGen 3.2 or later.

1. If it is not running, start the MashZone NextGen Server for the MashZone NextGen Repository with the global MashZone NextGen attributes that you wish to export. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

   ```
   padmin exportGlobalAttribs -f output-file [-l prestoURL]
   -u username -w password [-v]
   ```

   - *-f output-file* : is the path and name of the export file to hold the metadata.
   - *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
   - *-w password* : is the MashZone NextGen password to log in with.
   - *-v*: is an optional flag to turn on verbose logging.

   All messages and errors from the export process are sent to the command window (stdout). Once the export command completes successfully, you can use the output file to import mashables or mashups to another MashZone NextGen Repository.

Exporting Users, User Metadata and Groups

Yyou can export all users, user groups, user group assignments and MashZone NextGen User Attributes from the MashZone NextGen Repository to an export file. You can then import this file to another MashZone NextGen Repository.

Important: If you have configured MashZone NextGen to work with your LDAP Directory, this command *only* exports MashZone NextGen User Attributes. Data for users, user groups and user group assignments resides in LDAP.

This command is available only in MashZone NextGen 3.2 or later.

1. If it is not running, start the MashZone NextGen Server for the MashZone NextGen Repository with the user groups that you wish to export. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin exportUsersRoles -f output-file [-l prestoURL]
-u username -w password [-v]
```

   - *-f output-file* : is the path and name of the export file to hold the metadata.
   - *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
   - *-w password* : is the MashZone NextGen password to log in with.
   - *-v*: is an optional flag to turn on verbose logging.

   All messages and errors from the export process are sent to the command window (stdout). Once the export command completes successfully, you can use the output file to import mashables or mashups to another MashZone NextGen Repository.

Importing Mashable or Mashup MetaData

you must have a mashable or mashup export file to import. See "Exporting Mashable and Mashup MetaData" on page 1755 for instructions.

If the export file contains database mashables or mashups that use named datasources, you or a MashZone NextGen administrator must also define these datasources and drivers before importing these artifacts. These datasources and drivers *must exactly* match the datasources and drivers from the MashZone NextGen Server that was the source of these mashables or mashups.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.

2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin importServices -f input-file [-l prestoURL]
-u username -w password [-c] [-o] [-v]
```

   - *-f input-file* : is the path and name of the export file to import data from.
   - *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
   - *-w password* : is the MashZone NextGen password to log in with.
   - *-c*: is an optional flag to allow the import process to continue if errors occur when invoking mashables during the import. This flag does not force the import process to continue for other types of errors, such as network or server failures.

     By default, any import errors stop all further processing.
   - *-o*: is an optional flag to allow import information for a mashable or mashup to overwrite an existing mashable or mashup with the same ID.
   - *-v*: is an optional flag to turn on verbose logging.

   Messages and errors from the import process are sent to the console window (stdout).Once the import is successfully finished, you may confirm that mashables and mashups have been imported in MashZone NextGen Hub.

Example

The following example, imports data from a file named `localRestSvcs.xml`.

```
padmin importServices -f localRESTSvcs.xml -u Administrator -w manage
```

The following example from a Windows environment, imports data from a file named `localRestSvcs.xml` and logs all messages or errors from the import process to a file named `localRestImport.log`.

```
c:\MashZone NextGenversion\prestocli> padmin importServices
  -f localRESTSvcs.xml -u Administrator -w manage >> localRestImport.log
```

Importing Macros

you must have a macro export file to import. See "Exporting Macros" on page 1756 for instructions.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin importEmmlMacro -f input-file [-l prestoURL]
-u username -w password [-c] [-o] [-v]
```

- *-f input-file* : is the path and name of the export file to import data from.
- *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
- *-p password* : is the MashZone NextGen password to log in with.
- *-c*: is an optional flag to allow the import process to continue if errors occur when invoking mashables during the import. This flag does not force the import process to continue for other types of errors, such as network or server failures.

  By default, any import errors stop all further processing.
- *-o*: is an optional flag to allow import information for a mashable or mashup to overwrite an existing mashable or mashup with the same ID.
- *-v*: is an optional flag to turn on verbose logging.

Messages and errors from the import process are sent to the console window (stdout).Once the import is successfully finished, you may confirm that macros have been imported in MashZone NextGen Hub.

Importing App Metadata

you must have an app export file to import. See "Exporting App MetaData" on page 1758 for instructions.

If you choose to import dependent mashables or mashups that exist in the export file, you must also define any datasources and drivers used by these mashables or mashups. See "Importing Mashable or Mashup MetaData" on page 1763 for more information.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin importApps -f input-file [-q filter]
[-s -l prestoURL] -u username -w password
[-c] [-o] [-v]
```

- *-f input-file* : is the path and name of the export file to import data from. This may be an XML file or a ZIP file.
- *-q filter* : defines which apps to import from the export file. The filter can be:
  - `"all"` to import all apps from this export file. You can also use `"ALL"` or `"*"`.

- "`author=`*list-of-owner-ids* ", the user ID for a specific user or a list of comma-separated user IDs. This imports apps created by those users.
- "`category=`*list-of-categories* ", a specific category or a list of comma-separated categories. This imports apps with those categories. Apps with no category are not included.
- "`ids=`*list-of-app-ids* ", a specific app ID or a list of comma-separated app IDs. This imports those specific apps.
- "`name=`*app-name* ", the name of a specific app or a list of comma-separated names. This imports apps with those names.
- "`provider=`*list-of-providers* ", a specific provider or a list of comma-separated providers. This imports apps with those providers. Apps with no provider are not included.
- "`tag=`*list-of-tags* " , a specific user-defined tag or a list of comma-separated tags. This imports apps with those tags. Apps with no tags are not included.

- *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
- *-w password* : is the MashZone NextGen password to log in with.
- *-c*: is an optional flag to allow the import process to continue after import errors. By default, any import errors stop all further processing.
- *-o*: is an optional flag to allow import information for an app to overwrite an existing app with the same ID.
- *-s*: is an optional flag to import *all* the dependent mashables or mashups in the import file. Note that imports for dependent artifacts is not affected by any filtering.
- *-v*: is an optional flag to turn on verbose logging.

Messages and errors from the import process are sent to the console window (stdout).Once the import is successfully finished, you may confirm that apps have been imported in MashZone NextGen Hub.

Example

The following example, imports all the apps from a file named `MyorgApps.xml`.

```
padmin importApps -f MyorgApps.xml -u Administrator -w manage
```

The following example from a Windows environment, imports data from a file named `MyorgApps.xml` and logs all messages or errors from the import process to a file named `AppImport.log`.

```
c:\Program Files\JackBe\version\prestocli> padmin importApps
  -f MyorgApps.xml -u Administrator -w manage >> AppImport.log
```

Importing Pluggable Views or Libraries

you must have either:

- A library export file to import containing pluggable views or libraries from another MashZone NextGen Server. See "Exporting Pluggable Views or Libraries" on page 1759 for instructions.
- A directory containing resources for one pluggable view or pluggable library to import or update, plus an optional configuration file to provide additional metadata. See "Creating Pluggable Views or Libraries" on page 1045 for details.

You may use this command to deploy pluggable views and pluggable libraries that you have exported from one MashZone NextGen Server to another MashZone NextGen Server. See "Example 156. Example" on page 1768 for an example of this usage.

You may also use this command to add to or update pluggable views or pluggable libraries MashZone NextGen based on source files on your computer. See "Creating Pluggable Views or Libraries" on page 1045 for more information and examples.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command or terminal window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin importLib [-d input-directory] [-f input-file]
[-q filter] [-o] [-c] [-l prestoURL] -u username
-w password [-v]
```

   - *-d input-directory* : the root directory containing configuration and resources for one pluggable view or pluggable library to add to or update in MashZone NextGen.

     Note: You must include either the `-d` or the `-f` option

     See "Views and Libraries in MashZone NextGen" on page 1045 for more information on valid resources in this directory.

   - *-f input-file* : the path and name of the export ZIP file to use to import views and libraries from another MashZone NextGen Server.

     Note: You must include either the `-d` or the `-f` option.

   - *-q filter* : defines which pluggable views and libraries to import from the ZIP input file specified in the `-f` option. This option is not valid with the `-d` option.

     The filter can be:

     - `"all"` to import all pluggable views and pluggable libraries from this input file. You can also use `"ALL"` or `"*"`.

- "`author=`*list-of-owner-ids* ", the user ID for a specific user or a list of comma-separated user IDs. This imports the pluggable views and pluggable libraries listed in the input file that are owned by those users.
- "`mine`", to import all pluggable views or libraries from the input file that are owned by the user identified by the credentials used to execute this command (in the `-u` option).
- "`ids=`*list-of-app-ids* ", the ID for a specific pluggable view or pluggable library or a list of comma-separated view or library IDs. This imports those specific views or libraries from the input file.
- "`name=`*list-of-library-names* ", a specific pluggable view or library name.
- "`type=view`", to import only pluggable views from the input file.
- "`subtype=`*view-category-name* ", to import only pluggable views that belong to the specified view category from the input file.

- *-c*: an optional flag to allow the import process to continue after import errors. By default, any import errors stop all further processing.
- *-o*: an optional flag to allow import information for a view or library to overwrite an existing view or library with the same ID.
- *-l prestoUrl* : an optional URL to the MashZone NextGen Server where these views and libraries should be imported to. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.

  Use this option if the MashZone NextGen Server is remote, if it is not running in Tomcat or if it is not using the default Tomcat port.
- *-u username* : the MashZone NextGen username to use as credentials to execute this command. With the `-d` option, this also becomes the owner of the pluggable view or library that is imported or updated unless the properties file for this import specifies an owner.
- *-w password* : the MashZone NextGen password to use as credentials to execute this command.
- *-v*: an optional flag to turn on verbose logging.

Messages and errors from the import process are sent to the console window (stdout).

Example

The following example from a Windows environment, imports all pluggable views and libraries from the import file `localCustomViews.zip` to the local MashZone NextGen Server.

```
c:\MashZone NextGenversion\prestocli> padmin importLib
  -q "all" -f localCustomLibs.zip -u Administrator -w manage
```

The following example from a Windows environment, imports only the pluggable views in the view category `network` from the import file `remoteCustomViews.zip` to the local MashZone NextGen Server.

```
c:\MashZone NextGenversion\prestocli> padmin importLib
  -q "subtype=network" -f remoteCustomLibs.zip -u Administrator -w manage
```

For examples of adding pluggable views or libraries with the importLib command, see "Import Pluggable Library/View Examples" on page 1076 and "Managing Updates and Library Versions" on page 1085.

Importing MashZone NextGen Global Attributes

you must have a MashZone NextGen Global Attribute export file to import. See "Exporting MashZone NextGen Global Attributes" on page 1762 for instructions.

Note: This command is available only in MashZone NextGen 3.2 or later.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

   ```
   padmin importGlobalAttribs -f input-file [-l prestoURL]
   -u username -w password [-c] [-o] [-v]
   ```

   - *-f input-file* : is the path and name of the export file to import data from.
   - *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
   - *-p password* : is the MashZone NextGen password to log in with.
   - *-c*: is an optional flag to allow the import process to continue if errors occur during the import. By default, any import errors stop all further processing.
   - *-o*: is an optional flag to allow import information for a MashZone NextGen Global Attribute to overwrite an existing MashZone NextGen global attribute with the same ID.
   - *-v*: is an optional flag to turn on verbose logging.

   Messages and errors from the import process are sent to the console window (stdout).Once the import is successfully finished, you may confirm that MashZone NextGen Global Attributes have been imported in MashZone NextGen Hub.

Importing Users, User Metadata and Groups

you must have a users export file to import. See "Exporting Users, User Metadata and Groups" on page 1763 for instructions.

Note: This command is available only in MashZone NextGen 3.2 or later.

A user export file contains MashZone NextGen User Attributes. It may also contain users, user groups and user group assignments if you are using the default MashZone NextGen User Repository rather than an LDAP Directory.

Important: If you import users, you need to notify *all* the users included in the import that their password in the target MashZone NextGen Server has been changed to `welcome`.

1. If it is not started, start the MashZone NextGen Server for the MashZone NextGen Repository where you wish to import data. See "Start and Stop the MashZone NextGen Server" on page 1570 for instructions.
2. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
3. Enter this command:

```
padmin importUsersRoles -f input-file [-l prestoURL]
-u username -w password [-c] [-o] [-v]
```

   - *-f input-file* : is the path and name of the export file to import data from.
   - *-l prestoUrl* : is optional. Use this if the MashZone NextGen Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the MashZone NextGen username to log in with. This account *must* have MashZone NextGen administrator permissions.
   - *-p password* : is the MashZone NextGen password to log in with.
   - *-c*: is an optional flag to allow the import process to continue if errors occur during the import. By default, any import errors stop all further processing.
   - *-o*: is an optional flag to allow import information for a MashZone NextGen global attribute to overwrite an existing user, group, user group assignments or MashZone NextGen User Attribute with the same ID.
   - *-v*: is an optional flag to turn on verbose logging.

   Messages and errors from the import process are sent to the console window (stdout).Once the import is successfully finished, you may confirm that the appropriate data has been imported in MashZone NextGen Hub.

Deploying Multiple MashZone NextGen Servers in One Host

You can deploy several different, independent MashZone NextGen Servers on a single host. Each MashZone NextGen Server must be hosted in its own application server and have its own MashZone NextGen Repository.

- To host multiple, independent servers, simply install each *being sure* to change the ports assigned to each MashZone NextGen Server, MashZone NextGen Repository and the administration port for Tomcat.

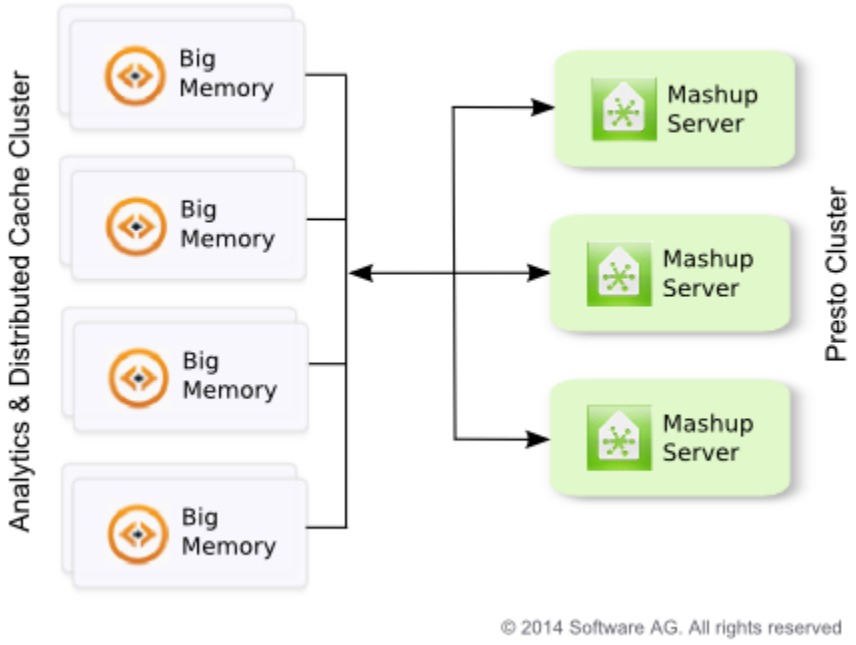
> **Note:** You can also create clusters of MashZone NextGen Servers to provide load balancing. See "Clustering MashZone NextGen Servers" on page 1771 for information.

Clustering MashZone NextGen Servers

In production environments, it is common to use clustering solutions to provide better performance for various loads, to provide high availability or to provide both. Because MashZone NextGen is a web application, using an HTTP session based on J2EE standards, you can apply the same cluster architectures and solutions to MashZone NextGen that you use with other web applications.

> **Note:** Clustering is *not* supported for MashZone NextGen if you also deploy Software AG MashZone or you use the Integrated MashZone Server with MashZone NextGen.

A common architecture for MashZone NextGen clusters looks something like this:



See "Setting Up a New Cluster" on page 1772 or "Adding New Members to an Existing Cluster" on page 1773 for the tasks you need to complete.

Setting Up a New Cluster

The configuration and deployment of a new cluster requires these basic steps:

- "Setting Up an External MashZone NextGen Configuration Folder" on page 1777: this allows you to keep most of the configuration and extensions for MashZone NextGen in a single set of folders that can be shared across the entire cluster. This simplifies both the initial configuration as well as ongoing updates and deployment of new mashables, mashups or apps.

  Note: This step is highly recommended, but not required. If you do not use a shared configuration folder, all subsequent updates to configuration or extensions for new artifacts must be manually copied to each member of the cluster.

  This folder should reside in a file system that is shared or mounted across the cluster. You may also need to provide data redundancy or failover capabilities for this shared file system.

  As part of this step, you also typically deploy one MashZone NextGen Server in the cluster and complete most of the basic configuration that will be shared across the cluster.

- "Sharing the MashZone NextGen Repository in Clustered Environments" on page 1774: all nodes in the cluster work with a shared MashZone NextGen Repository which you must create and configure.

  Sharing the MashZone NextGen Repository does not, by itself, provide any data redundancy, load balancing or failover capabilities for the database. These requirements are handled in the data layer by your database server or other replication/synchronization solutions, such as DRBD. For more information, see documentation for your database or replication/synchronization solution.

- *Configuring Caching for the Cluster*: each MashZone NextGen Server has a local cache for mashable and mashup responses as well as local caches for updates to artifacts. If MashZone NextGen Analytics is enabled in your MashZone NextGen license, the MashZone NextGen Analytics In-Memory Stores are also local.

  In clusters you:

  - Can leave the response cache as a local cache or you can configure a distributed cache that all MashZone NextGen Servers in the cluster share.
  - *Must* configure a distributed cache for artifact updates that all MashZone NextGen Servers in the cluster share.
  - *Must* configure a distributed cache for the MashZone NextGen Analytics In-Memory Stores that all MashZone NextGen Servers in the cluster share.

See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for instructions on how to configure BigMemory, or other caching solutions, as a distributed cache for .MashZone NextGen

- *Defining the Application Server Cluster*: the application servers that host each MashZone NextGen Server define and handle clustering requirements at the application layer. You can also add a load balancer to the cluster.

  In addition to the basic cluster configuration required by your application server and load balancer, MashZone NextGen has a single requirement for application-layer cluster configuration. You must either:

  - Enable session replication in each application server in the cluster.
  - Enable session affinity, sometimes also called 'sticky sessions,' in the load balancer.
  - Or do both.

  See documentation for your application server and/or load balancer for information on how to do this.

- *Adding Additional MashZone NextGen Servers to the Cluster*: once you have set up the shared resources, you can deploy and add additional members to the cluster. See "Adding New Members to an Existing Cluster" on page 1773 for instructions.

- *Add MetaData and Deploy Artifacts*: for this new environment. For artifacts, you can automate some parts of this process using export and import commands. See "Deploying MashZone NextGen Artifacts and Other Metadata" on page 1750 for instructions.

Adding New Members to an Existing Cluster

To add additional MashZone NextGen Servers to an existing cluster

1. Install the MashZone NextGen Server. See *Installing Software AG Products* for instructions.
2. Configure the MashZone NextGen Server to use the shared MashZone NextGen Repository for the cluster. See "Share an Existing MashZone NextGen Repository" on page 1776 for instructions.
3. If the cluster has a shared external configuration folder, add this folder and any subfolders to the classpath for the MashZone NextGen Server's application server to enable access to this shared configuration.

   Depending on your application server, you may update the classpath in the administration console, in configuration files or in the startup script for the application server. See documentation for your application server for more information.
4. If the cluster does not have a shared external configuration folder, copy the configuration and extension files from an existing MashZone NextGen Server in the cluster to the new MashZone NextGen Server.

See "MashZone NextGen File-Based Configuration and Extensions" on page 1778 for a list of files and folders to copy. Be sure to also copy the JDBC drivers used for database mashables and mashups. See "Sharing JDBC Drivers in Clustered Environments" on page 1783 for more information.

5. Copy the server configuration that cannot be shared from an existing MashZone NextGen Server in the cluster to the new MashZone NextGen Server. See "MashZone NextGen File-Based Configuration and Extensions" on page 1778 for details on the files and locations for this step.
6. Update the application server that hosts the new MashZone NextGen Server with the same cluster configuration as other cluster members.

   In addition to the basic cluster configuration required by your application server and load balancer, MashZone NextGen has a single requirement for application-layer cluster configuration. You must either:

   - Enable session replication in each application server in the cluster.
   - Enable session affinity, sometimes also called 'sticky sessions,' in the load balancer.
   - Or do both.

   See documentation for your application server and/or load balancer for information on how to do this.
7. Restart the new MashZone NextGen Server.

Sharing the MashZone NextGen Repository in Clustered Environments

In clustered environments, all MashZone NextGen Servers in the cluster must work with a single, shared MashZone NextGen Repository. You can "Create and Share a New MashZone NextGen Repository" on page 1774 with cluster members, typically when you are creating new environments. Or you can "Share an Existing MashZone NextGen Repository" on page 1776 within a cluster.

Create and Share a New MashZone NextGen Repository

To create a new shared repository

1. Create a new MashZone NextGen Repository in the appropriate database for your environment.

   Using the SQL tool for the database that will be host, add MashZone NextGen Repository tables with the scripts shown below from the corresponding folder in *MashZoneNG-install* /prestorepository:

Database	Folder	SQL Scripts
Microsoft SQL Server	mssqldb	■ createDBTables.txt for MetaData and the default User Repository ■ createSnapsTables.txt for Snapshots ■ createSchedulerTables.txt for Scheduler
MySQL	mysqldb	
Oracle	oracledb	
PostgreSQL	postgresdb	

There are also scripts to drop the corresponding MashZone NextGen Repository tables in these folders, if needed.

2. Copy the JAR file for the JDBC driver for your database to the *MashZoneNG-config* folder for JAR files.

   If you are using a shared external configuration folder, you can copy the JAR to that single folder. For example, you might copy this to G:\PrestoConfig\lib on a Windows system or /mountA/home/PrestoConfig/lib on Linux, Mac OS X or UNIX systems.

   If you are not sharing a configuration folder across the cluster, you must copy the JAR to *web-apps-home* /presto/WEB-INF/lib for each MashZone NextGen Server that uses this MashZone NextGen Repository. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

3. If this is a new cluster, update configuration information for the MetaData, User and Snapshot repositories for one MashZone NextGen Server in the cluster. See steps 3 onward in "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for instructions.

4. Copy the rdsJdbc.properties file to the *MashZoneNG-config* folder for the entire cluster or to each MashZone NextGen Server in the cluster.

   If you are using a shared external MashZone NextGen configuration folder, both of these files are already in the list of property files that you place in this folder. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

5. Enable distributed caching for artifacts (required) and optionally distributed caching for mashable/mashup responses. See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for more information and instructions.

6. If the MashZone NextGen Repository is hosted in Microsoft SQL Server, MySQL or Oracle, change the repository JAR in the MashZone NextGen Server.

7. Restart each MashZone NextGen Server in the cluster.

Share an Existing MashZone NextGen Repository

If you are creating a cluster using an existing MashZone NextGen Repository or simply adding members to an existing cluster, you simply update each new MashZone NextGen Server in the cluster to use the existing repository.

1. If the cluster does *not* have a shared JDBC driver folder and a shared external configuration folder:

   a. Copy the JAR file for the JDBC driver for your database to the *web-apps-home* /presto/WEB-INF/lib folder for the new MashZone NextGen Server cluster member.

   b. Copy the rdsJdbc.properties file from the *web-apps* /presto/WEB-INF/classes folder of an existing MashZone NextGen Server in the cluster, to the same folder for the new cluster member.

   See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information on shared configuration for clusters.

2. If the cluster does have a shared JDBC driver folder and a shared external configuration folder, update the application server's classpath for the new cluster member to include all three folders.

   Depending on your application server, you may update the classpath in the administration console, in configuration files or in the startup script for the application server. See documentation for your application server for more information.

   For Tomcat on Windows, for example, you can edit the *tomcat-install* /bin/catalina.bat file and add the path to this folder to the classpath environmental variable to be something like this:

   ```
   set "CLASSPATH=%CLASSPATH%;C:\PrestoConfig;C:\PrestoConfig\classes;C:
   \PrestoConfig\lib;C:\PrestoConfig\db\jdbc"
   ```

   On Linux, Mac OS X or UNIX systems, you would update *tomcat-install* /bin/catalina.sh to something like this:

   ```
   CLASSPATH="$CLASSPATH":/users/PrestoConfig:/users/PrestoConfig/
   classes:/users/PrestoConfig/lib:users/PrestoConfig/db/jdbc
   ```

3. Enable distributed caching for artifacts (required) and optionally distributed caching for mashable/mashup responses. See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for more information and instructions.

4. If the MashZone NextGen Repository is hosted in Microsoft SQL Server, MySQL or Oracle, change the repository JAR in the MashZone NextGen Server.

5. Restart the new MashZone NextGen Server for this cluster.

Setting Up an External MashZone NextGen Configuration Folder

Most configuration for MashZone NextGen and most of the extensions that you add for your organization's use are stored in the MashZone NextGen Repository. However, some MashZone NextGen configuration and extensions are file based.

By default, MashZone NextGen keeps configuration and extensions in the MashZone NextGen Server web application in these folders:

- *MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/classes for class, configuration and extension files
- *MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/lib and *MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/config for JAR files.

You can move most of these configuration and extension files to folders that are external to the MashZone NextGen Server.

Important: MashZone NextGen documentation refers to all of these folders as *MashZoneNG-config* .

Using external configuration folders for MashZone NextGen is a best practice as they simplify deployment and upgrades of the MashZone NextGen Server. They also simplify configuration management for clustered environments. External configuration folders are not required, however.

To create and use an external configuration folder for MashZone NextGen

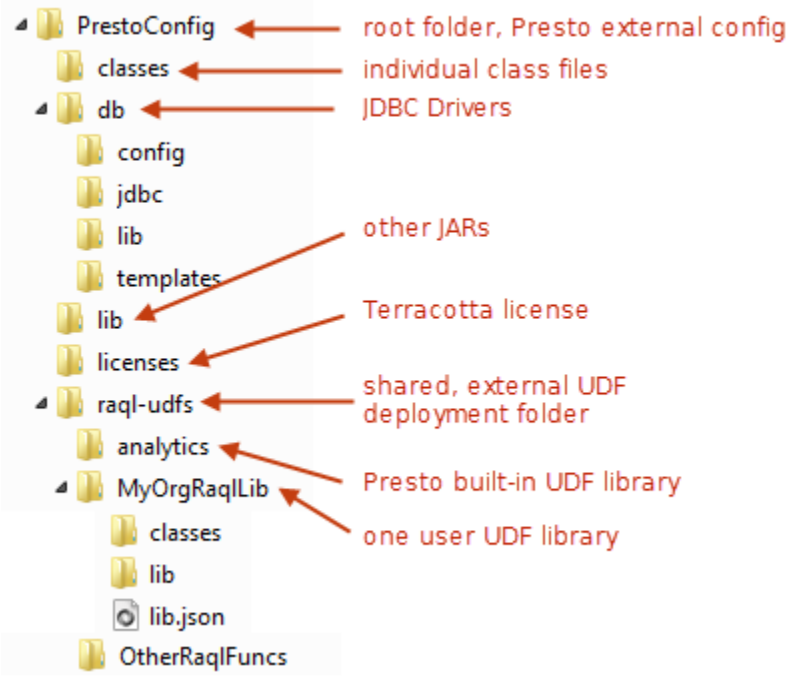
1. Create the top-level external folder to use for MashZone NextGen configuration, such as PrestoConfig. In clustered environments, share or mount this folder across the entire cluster.

   You can create subfolders under this external folder to organize configuration and extensions.

2. For clustered environments, create subfolders under the top-level external configuration folder for:

   - The standard classes and lib folders.
   - JDBC drivers for database mashables and mashups for the cluster. If desired, you can create an external folder for JDBC drivers in non-clustered environments. See "Sharing JDBC Drivers in Clustered Environments" on page 1783 for instructions.
   - Built-in and user-defined functions for use in RAQL queries for MashZone NextGen Analytics. See "Configure, Compile, Deploy and Test User-Defined Functions" on page 807 for more information.

   The external configuration folder tree should now look something like this:



3. If not complete, finish configuration for the MashZone NextGen Server and move the configuration and extension files to the external configuration folder or an appropriate subfolder. See the "MashZone NextGen File-Based Configuration and Extensions" on page 1778 section for the specific configuration steps, files and locations.

4. Add the external MashZone NextGen configuration folder, *and any subfolder* that contains extensions or JAR files, to the classpath for the application server(s) hosting the MashZone NextGen Server.

   You may update the classpath in configuration files or in the startup script for the application server.

   For Windows environments, for example, you can edit the *tomee-install* /bin/setenv.bat file and update the classpath environmental variable to be something like this:

   ```
   set "CLASSPATH=%CLASSPATH%;C:\PrestoConfig;C:\PrestoConfig\classes;C:
   \PrestoConfig\lib;C:\PrestoConfig\db\jdbc"
   ```

   On Linux, Mac OS X or UNIX systems, you would update *tomee-install* /bin/setenv.sh to something like this:

   ```
   CLASSPATH="$CLASSPATH":/users/PrestoConfig:/users/PrestoConfig/
   classes:/users/PrestoConfig/lib:users/PrestoConfig/db/jdbc
   ```

MashZone NextGen File-Based Configuration and Extensions

Most file-based configuration or extensions involve information that MashZone NextGen needs to connect to the MashZone NextGen Repository or extensions that

must be added to the application server's classpath. In clustered environments, you can share extensions and some of this file-based configuration using an external configuration folder. See "MashZone NextGen Configuration Files That Can Be External" on page 1779 and "MashZone NextGen Extensions" on page 1782 for details on resources that can be shared across a cluster.

Some file-based configuration, however, *must* reside in the web application for each MashZone NextGen Server. In clusters, this configuration must be replicated in each cluster member. See "MashZone NextGen Configuration Files That Must Be Internal " on page 1780 for details.

MashZone NextGen Configuration Files That Can Be External

File	Description and Configuration	Default Location
dynamiccache.xml	Default configuration information for dynamic In-Memory Stores created by MashZone NextGen Analytics.	*MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/classes
ehcache.xml	Configuration information for MashZone NextGen caches. This also contains configuration for MashZone NextGen Analytics In-Memory Stores from version 3.6.	
presto.config	Miscellaneous MashZone NextGen properties, including the path to the deployed web app home folder and the path, if used, for shared JDBC drivers. See "Sharing JDBC Drivers in Clustered Environments" on page 1783 for more information.	
rdsJdbc.properties	Connection information for the MetaData and User section of the MashZone NextGen Repository. **Note:** Although MashZone NextGen uses JNDI to connect to the MashZone NextGen Repository, JDBC connection	

File	Description and Configuration	Default Location
	properties are also used in some specific cases. See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for details.	
The Terracotta BigMemory license file	The license file for BigMemory, used for MashZone NextGen caches and MashZone NextGen AnalyticsIn-Memory Stores, is a separate license file from the MashZone NextGen license. You can keep the BigMemory license in an external folder shared across the cluster. See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for required configuration steps to enable a shared license.	
userRespositoryLdap.properties	Connection information for your LDAP Directory. See "Integrate Your LDAP Directory with MashZone NextGen" on page 1590 for details.	

MashZone NextGen Configuration Files That Must Be Internal

The file-based configuration that must remain in each MashZone NextGen Server web application resides in the *web-apps-home* /presto/WEB-INF/classes folder.

For upgrades to new MashZone NextGen versions, you can generally copy these configuration files from your existing MashZone NextGen version to the new version. Review the MashZone NextGen Release Notes for changes or new features that may require updates to configuration.

For clustered environments, you *must* copy these configuration files to each cluster member. In most cases, you change configuration once, when you first deploy a MashZone NextGen Server in the cluster. Any subsequent changes to this configuration

for one cluster member, however, must be copied to all other cluster members manually, using a scheduled job or using another replication scheme.

File	Description and Configuration
applicationContext-commonServices.xml	You edit configuration in this file if you choose to use distributed response caching for MashZone NextGen. See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 1664 for more information. You may need to update this configuration, as needed, to add additional distributed cache nodes to tune performance.
applicationContext-security.xml	You edit this file initially to enable either SSO authentication or X509 certificate authentication for MashZone NextGen. See "Authentication with Single Sign-On Solutions" on page 1607 or "Authentication with Digital Certificates/SSL" on page 1614 for more information.
applicationContext-security-x509.xml	You edit this file initially to enable X509 certificate authentication for MashZone NextGen. See "Authentication with Digital Certificates/SSL" on page 1614 for more information.
applicationContext-scheduler.xml	You edit this file when you move the MashZone NextGen Repository from the default Derby database to a robust solution. See "Move the MashZone NextGen and MashZone Repositories to a Robust Database Solution" on page 1574 for more information.
log4j.properties	This file is updated automatically when you change logging configuration in the Admin Console. See "Configure Logging for the

File	Description and Configuration
	MashZone NextGen Server" on page 1650 for details. When you change logging for MashZone NextGen Servers in a cluster, only the specific MashZone NextGen Server that the Admin Console is connected to is affected. To change logging for the entire cluster, you must update this file and copy it to each cluster member.
userRepositoryApplicationContext.xml userRepositoryApplicationContext-ldap.xml	You edit these files when you configure MashZone NextGen to use your LDAP Directory as the user repository. See "Integrate Your LDAP Directory with MashZone NextGen" on page 1590 for details.

MashZone NextGen Extensions

Some extensions, such as macros, are registered and reside in the MashZone NextGen Repository. Any of the following file-based extensions can reside in an external folder:

File	Default Location
JAR files for JDBC drivers for datasources used with database mashables or with mashups. To share JDBC drivers across a cluster of MashZone NextGen Servers, you must set up a shared JDBC folder. See "Sharing JDBC Drivers in Clustered Environments" on page 1783 for the specific details. If you do not create a shared folder, then you *must* copy the JARs for all JDBC drivers to the default location for each MashZone NextGen Server in the cluster.	*MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/config/db/jdbc
Scripts or classes called in mashups in EMML using the <script> statement. This includes: ■ JavaScript files ■ Any Java class that is not in `java.lang`	*MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/classes or

File	Default Location
■ JRuby scripts ■ Groovy scripts See "Adding User-Defined Scripting Code to Mashups" on page 524 for details.	*MashZoneNG-install* /apache-tomee-jaxrs/presto/WEB-INF/lib (for JARs)
XSLT stylesheets called in mashups in EMML using the <xslt> statement. See "<xslt>" on page 529 for details.	
Custom XPath function classes used in mashups in EMML. See "Defining Custom XPath Functions" on page 601 for details.	
Local copies of WSDL files used for WSDL web services.	
Custom security profile classes used with mashables. See "Configure Secure Connections for Mashables" on page 164 for details.	
Custom certificate validation classes for certificate authentication. See "Configure Additional Certificate Validation" on page 1618 for details.	
Custom filter classes for single sign-on authentication. See "Implementing a Custom SSO Filter" on page 1614 for details.	
Classes and third-party libraries for a user-defined function library to use with RAQL. See "Create and Add User-Defined Functions for RAQL Queries" on page 804 for more information.	

Sharing JDBC Drivers in Clustered Environments

In clustered environments, each cluster member must have access to the JDBC drivers that are used by database mashables or mashups. You can create a folder for JDBC driver JARs that is external to all MashZone NextGen Servers in the cluster. This external folder can be shared with all cluster members to ensure access. Once you have set up a shared

driver folder, any JDBC drivers that you add in the Admin Console are immediately available to all cluster members.

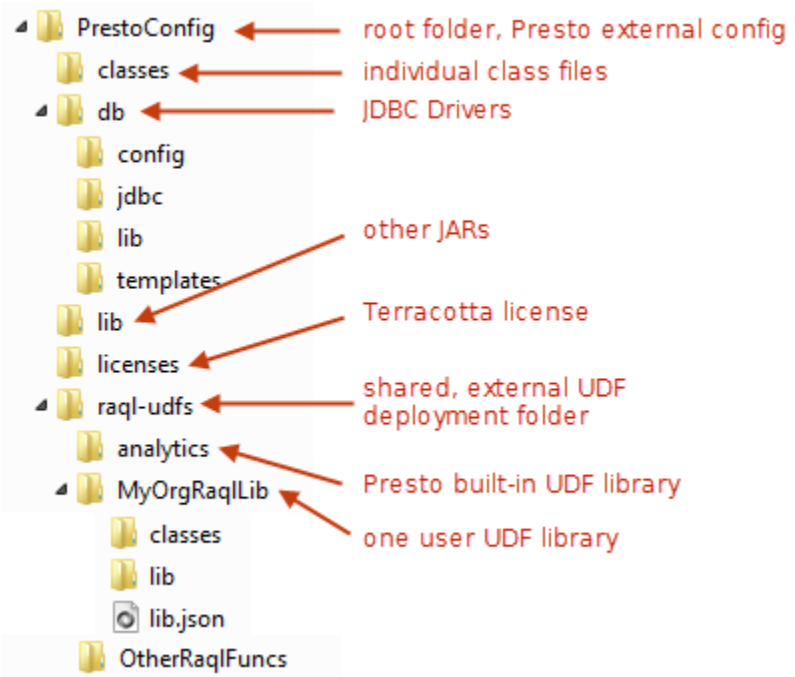
Note: If you do not set up a shared JDBC driver folder, you must copy the JARs for each driver, as they are added, to the default driver location for each MashZone NextGen Server in the cluster.

1. Create a folder for JDBC drivers that is external to the MashZone NextGen Server web application. In clustered environments, this folder should reside in a file system that is shared or mounted across the entire cluster.

   If you are using an external configuration folder for MashZone NextGen, it is a best practice to add the JDBC driver folder there. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

2. Copy the entire *web-apps-home* /presto/config/db folder from one MashZone NextGen Server in the cluster to the external shared folder for drivers created in the previous step.

   Your shared folder tree should look something like this:



3. In any text editor, edit the *MashZoneNG-config* /presto.config file.

   Note: The *MashZoneNG-config* folder may be in the default location or in an external location. See "Setting Up an External MashZone NextGen Configuration Folder" on page 1777 for more information.

   a. Set the `codegen.absolutepath.config` property to point to the external JDBC driver folder.

For a Windows system with the external JDBC driver folder on a shared drive G:\PrestoConfig\db, for example, the property would look like this:

codegen.absolutepath.config = g:\\PrestoConfig

For Linux, Mac OS X or UNIX systems with an external folder of /mountA/home/PrestoConfig/db, it might look something like this:

codegen.absolutepath.config = /mountA/home/PrestoConfig

b. Save the changes to this file.

4. Restart the entire cluster to apply these changes.

15 Additional Information and Support

Samples, Help and Other Documentation

You can find samples, help and other documentation for MashZone NextGen in:

- MashZone NextGen User and Developer Guide: You can download MashZone NextGen documentation at https://empower.softwareag.com/ for offline access. An Empower account is required.
- *Online Help*: is accessible in MashZone NextGen Hub and the AppDepot from *?* help buttons. Help is also available in the MashZone NextGen Add-On for Excel in the toolbar for the add-on.
- *Documentation for Other Software AG Products*: is available online at http://documentation.softwareag.com with an Empower or from http://techcommunity.softwareag.com/welcome-documentation with a Software AG Tech Communities account.
- *MashZone NextGenMashup and Macro Samples*: samples of mashups scripts and macro libraries using the EMML language are available in the Mashup Editor in MashZone NextGen Hub. Samples for RAQL can be found in the *MashZoneNG-install* /prestocli/raql-samples folder.

Version and License Information

You can find version and other system information for MashZone NextGen in the Admin Console.

For system information:

1. Log in to MashZone NextGen and click **Admin Console**.
2. Expand the Server menu section and click **Server Info**.
3. Select the **Platform Information** or **System Information** tabs.

For license information:

1. Log in to MashZone NextGen Add-On for Excel and click **Admin Console**.
2. Expand the Server menu section and click **License**.

Add-ons, Updates, Forums and Technical Support

Find, Contact or Download	From
The MashZone NextGen Add-On for Excel installation package	Software AG Tech Community for MashZone NextGen Add-On for Excel at http://techcommunity.softwareag.com/ecosystem/

Find, Contact or Download	From
The MashZone NextGen Add-On for Portals	communities/public/apama/products/presto/downloads.
Questions and discussion forums	http://techcommunity.softwareag.com/ecosystem/communities/public/apama/products/presto
The MashZone NextGen Add-On for SharePoint	Empower, the Software AG support portal, at http://empower.softwareag.com/.
Technical support, use the eService features.	To request an Empower account, send an email with your name, company name and company email address to mailto:empower@softwareag.com. For crisis incidents, please call 1-800-525-7859 in the United States. Outside the US, you can find your local access number in the Software AG Global Directory.