MashZone NextGen Administration Guide

Version 10.1

October 2017

This document applies to MashZone NextGen Version 10.1 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.

Document ID: JBP-UD-101-20171017

Table of Contents

Preface

The MashZone NextGen Administration Guide includes information for administrators to configure and manage MashZone NextGen.

1 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. See "Start and Stop the MashZone NextGen Server" on page 17 for instructions.
2. Login to MashZone NextGen:
   a. Open MashZone NextGen in a browser at `http://localhost:8080/mashzone`.

      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 repository to a robust database solution" on page 19 for instructions.
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 34 for instructions.
6. Configure the Event Service. See "Event Service Configuration and Administration" on page 169 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 and

BigMemory" on page 18 and "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 127 for instructions.

Additional MashZone NextGen System and Software Requirements

- For basic requirements to install MashZone NextGen, see the *System Requirements for Software AG Products* guide.

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 recommendations for MashZone NextGen:

- A robust, compatible database to host the MashZone NextGen Repository is required.

  Important: The MashZone NextGen repository is initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move the repositories to a robust and compatible solution. See *System Requirements for Software AG Products* 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.

- To ensure a secure connection between MashZone NextGen server and client it is recommended to operate your MashZone NextGen system via HTTPS as communication protocol. You can configure your application server accordingly after you have installed MashZone NextGen. See "Configure HTTPS and Certficate Stores in the Application Server" on page 110 for details.

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
rtbs.war	Event Service
ibo.war and ibo-config.war	IBO user interface

MashZone NextGen also installs the following additional software:

- Apache's Tomcat Servlet Container, version 8.5.15
- Derby Database, version 10.5.3.0.

Important: The MashZone NextGen repository is initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move the repositories to a robust and compatible solution. See "Move the MashZone NextGen repository to a robust database solution" on page 19 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:

- Workspace apps (dashboards) for business process monitoring from Optimize.

MashZone NextGen Installation Folders

```
SoftwareAG
- MashZoneNG
  - apache-tomcat (the container and web apps)
  - ibo-dashboards (for IBO + built-in dashboards)
  - mashzone (external resources for the Integrated Servers)
    - clientapps
    - 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

Most MashZone NextGen components 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 Event Service.

Start the MashZone NextGen Server

1. If the MashZone NextGen Repository has been moved from the default Derby database and they are *not* already running, manually start these databases following the instructions for their host database.
2. 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-tomcat/bin/startup.bat`

       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**.

       Note: Starting startup.bat from the file system using Windows Explorer does not work.
   - For Linux, Mac OS X or UNIX systems, open a new terminal window and move to this folder:

     `% cd` *`MashZoneNG-install`*`/apache-tomcat/bin`

     Then enter this command:

     `% startup.sh`

   This also automatically starts the Integrated 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 Mashup Editor, the App Editor and the Admin Console.

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-tomcat/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-tomcat/bin
     ```

     Then enter this command:

     ```
     % cataline.sh stop
     ```

   This also automatically stops the Integrated Event Service servers.
2. If the MashZone NextGen Repository has been moved from the default Derby database, you can also choose to stop this database. See documentation for the 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 and BigMemory

To use MashZone NextGen a license is required.

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 124 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 license. 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 MashZoneNGLicense.xml file into the *MashZoneNG-install* /apache-tomcat/conf folder.

   Note: If MashZone NextGen is deployed in a cluster, copy the license file to this folder in every cluster member.

3. If a BigMemory license is required:
   a. Copy the license file terracotta.key to the *MashZoneNG-install* /apache-tomcat/conf folder.

      Note: If MashZone NextGen is deployed in a cluster, you must copy this file to every cluster member.

   b. Add the following Java system property to the MashZone NextGen server configuration file <*MashZone NextGen installation*> /apache-tomcat/conf/wrapper.conf:

      ```
      wrapper.java.additional.<n+1>=-Dcom.tc.productkey.path=MashZoneNG-install/apache-tomcat/conf/terracotta.key
      ```

      Where n is the number of last additional Java parameter.

      Note: If MashZone NextGen is deployed in a cluster, you must update the server configuration files for every cluster member.

4. Restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 17 for instructions.

Move the MashZone NextGen repository to a robust database solution

The MashZone NextGen repository is initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move the repositories to a robust and compatible solution.

You can host the MashZone NextGen repository in any database that MashZone NextGen supports. See "Additional MashZone NextGen System and Software Requirements" on page 15 in System Requirements for details.

You can move the repository to one of the following databases:

- *Microsoft SQL Server*: see "Move MashZone NextGen repository to Microsoft SQL Server" on page 21 for instructions.
- *MySQL*: see "Move the MashZone NextGen repository to MySQL" on page 24 for instructions.
- *Oracle*: see "Move the MashZone NextGen repository to Oracle" on page 27 for instructions.
- *PostGres*: see "Move the MashZone NextGen repository to PostGres" on page 31 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 Tomcat, 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-tomcat/conf/context.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 repository 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 76 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. See "Additional MashZone NextGen System and Software Requirements" on page 15 for details.

     Note: The JTDS driver for SQL Server is the recommended driver for use with MashZone NextGen 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 case insensitive 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.
   - If you do not use the default **dbo** schema, you have to specify the name of the used schema (value="schema_name") in the bean PMF in the rdsApplicationContext.xml file.

     The file is located in <*MashZone NextGen installation*> \apache-tomcat\webapps \mashzone\WEB-INF\classes\.

     ```
     <bean id="pmf" class="com.jackbe.jbp.sas.rds.impl.jdo
                                .PersistenceManagerFactoryBean">
            <property name="lifecycleListener" ref="entityLifecycleLsnr"/>
            <property name="dataSource" ref="dataSource" />
            <property name="configLocation"
                           value="classpath:datanucleus.properties"/>
             <!-- overwrite settings in configLocation file -->
     ```

```
        <property name="jdoProperties">
<map>
<entry key="datanucleus.ConnectionFactoryName"
           value="java:comp/env/MashzoneNextGenRepository"/>
<entry key="datanucleus.storeManagerType" value="rdbms"/>
<entry key="datanucleus.mapping.Schema" value="schema_name"/>
</map>
        </property>
    </bean>
```

3. Start the database that will become host to the MashZone NextGen Repository, if it is 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:

   MashZoneNG-install /apache-tomcat/lib
6. Replace the JAR for the MashZone NextGen Repository:
   a. Remove the *web-apps-home* /mashzone/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* /mashzone/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* /mashzone/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-tomcat/conf/context.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
name="MashzoneNextGenRepository" auth="Container" type="javax.sql.DataSource"
      maxTotal="200" maxIdle="30" maxWaitMillis="10000"
      username="app" password="app" driverClassName="net.sourceforge.jtds.jdbc.Driver"
      url="jdbc:jtds:sqlserver://host-name:port/database"
/>
```

   The JTA managed property *must* be `false`.

   b. If needed, update optional properties. See Tomcat 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 Repository Connection Pool" on page 231.

10. 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 20 for suggestions.

11. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /mashzone/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 Tomcat.
    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.

12. Restart the MashZone NextGen Server to apply these changes.

    If the MashZone NextGen Server wedoes not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 20 for suggestions.

13. 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 148 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 145 for information on specific configuration properties.

f. Click **Save**.

14. Restart the MashZone NextGen Server to apply these changes.

15. 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/mashzone/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/mashzone/edge/api`

Move the MashZone NextGen repository 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 76 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 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. See "Additional MashZone NextGen System and Software Requirements" on page 15 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 that will become host to the MashZone NextGen Repository, if it is 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-tomcat/mashzone/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* /mashzone/WEB-INF/lib folder.
6. Copy the MySQL JDBC driver jar file to *MashZoneNG-install* /apache-tomcat/lib.
7. Open the *MashZoneNG-install* /apache-tomcat/conf/context.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:

```
name="MashzoneNextGenRepository" auth="Container" type="javax.sql.DataSource"
      maxTotal="200" maxIdle="30" maxWaitMillis="10000"
      username="app" password="app" driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://host-name/databasename"
/>
```

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 Tomcat 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 Repository Connection Pool" on page 231.

9. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /mashzone/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 Tomcat.

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.

10. Start the MashZone NextGen Server to apply these changes. This also starts the MashZone Server.

If the MashZone NextGen Server does not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 20 for suggestions.

11. 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 148 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 145 for information on specific configuration properties.

f. Click **Save**.

12. Restart the MashZone NextGen Server to apply these changes.

13. 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/mashzone/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/mashzone/edge/api`

Move the MashZone NextGen repository 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 76 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 in a new database, create the database following Oracle documentation.

   - Make sure this database is supported by MashZone NextGen. See "Additional MashZone NextGen System and Software Requirements" on page 15 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 that will become host to the MashZone NextGen Repository, if it is 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:
   - createDB.txt
   - 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* /mashzone/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* /mashzone/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:

   MashZoneNG-install /apache-tomcat/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: http://mvnrepository.com/artifact/org.quartz-scheduler/quartz-oracle/1.7.2.
      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-tomcat/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* /mashzone/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-tomcat/conf/context.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
name="MashzoneNextGenRepository" auth="Container" type="javax.sql.DataSource"
      maxTotal="200" maxIdle="30" maxWaitMillis="10000"
      username="app" password="app" driverClassName="oracle.jdbc.OracleDriver"
      url="jdbc:oracle:drivertype:@host-name:port:dbname"
/>
```

The JTA managed property *must* be `false`.

b. If needed, update optional properties. See Tomcat 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 Repository Connection Pool" on page 231.

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 20 for suggestions.

12. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /mashzone/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 Tomcat.
    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.

    If the MashZone NextGen Server wedoes not start up successfully, see "Troubleshooting Connections to the MashZone NextGen Repository" on page 20 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 148 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 145 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 -`	`./publish-global-macros -u Administrator -p manage -`

For Windows	For Linux, OS/X or UNIX
`url http://app-server:port/mashzone/edge/api`	`url http://app-server:port/mashzone/edge/api`

Move the MashZone NextGen repository 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 76 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. See "Additional MashZone NextGen System and Software Requirements" on page 15 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, you may need to specifically set the locale used by the database to ensure case-insensitive sorting.

3. Start the database that will become host to the MashZone NextGen Repository, if it is 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* /mashzone/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.PostgreSQLDelegate</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.

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: *MashZoneNG-install* /apache-tomcat/lib.

7. Open rdsApplicationContext.xml under *MashZoneNG-install* /apache-tomcat/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-tomcat/conf/context.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:

```
name="MashzoneNextGenRepository" auth="Container"
      type="javax.sql.DataSource"
      maxTotal="200" maxIdle="30" maxWaitMillis="10000"
      username="app" password="app"
      driverClassName="org.Postgresql.Driver"
      url="jdbc:postgresql://host-name:port/databasename"
/>
```

    The JTA managed property *must* be `false`.

    b. If needed, update optional properties. See Tomcat 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 Repository Connection Pool" on page 231.

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 20 for suggestions.

12. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /mashzone/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 Tomcat.
    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 20 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 148 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 145 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/mashzone/edge/api`	`./publish-global-macros -u Administrator -p manage -url http://app-server:port/mashzone/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 17 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 280 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 280 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 36.
   - Authentication mechanisms. See "Defining the Authentication Scheme" on page 37.
   - Authorization mechanisms. See "Defining the Authorization Scheme" on page 38.
   - 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 40.

   See also "Expose User Attributes from the User Repository in MashZone NextGen" on page 156 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 76 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 76 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 37, "Defining the Authorization Scheme" on page 38 and "Enabling

MashZone NextGen Application Queries for All LDAP Users or Groups for Permissions" on page 40) 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 74.

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,ou=system
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 Presto_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: The MashZone NextGen repository is initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move the 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 42
- "Manage User Groups" on page 43
- "Automatically Assign New Users to Groups" on page 44

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 42 and "Edit, Grant Permissions and other User Management Tasks" on page 43 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

Procedure

1. If needed, open the Admin Console.
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 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 44
- "Edit, Grant Permissions and other User Management Tasks" on page 43

Edit, Grant Permissions and other User Management Tasks

Procedure

1. If needed, open the Admin Console.
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 **Reset password** and enter a new password.
   - Click **Edit** to change the user'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 77 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 NextGen groups 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 77 for details.

Note: MashZone NextGen administrators and artifact owners automatically have permission to run all artifacts or the artifacts they own respectively.

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 79 for information.

1. If needed, open the Admin Console.
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: `Presto_Administrator`, `Presto_Developer`, `Presto_PowerUser`, `Presto_AuthenticatedUser` or `Presto_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.

Grant dashboard and data feed permissions via API console

You can use the API console to grand the permissions to create and edit dashboards and data feeds to MashZone NextGen users and user groups.

You have two options to assign the permissions to specific users and user groups. Using the Admin Console, as described in "Manage Users" on page 42 and "Manage User Groups" on page 43, and in addition, by using the API console.

By default, the permissions to create and edit dashboard and data feeds is only assigned to members of the **Presto_Developer** group, see "Built-In MashZone NextGen User Groups and Permissions" on page 77 for details. By using the API console you can assign the permissions required to other MashZone NextGen users.

Procedure

1. Open the API Console.
2. Enter the following code blocks if you want to assign the permissions to a specific user. Enter the relevant user name for the `userID` parameter.

```
{
    "version": "1.1",
    "sid": "PolicyService",
```

```
    "svcVersion": "0.1",
    "oid": "addPermissions",
    "params": [
        "Mashzone",
        "type.entity.uiObject",
        "accessDashboardEditor",
        [
            {
                "principalId": "userID",
                "principalTypeId": "User"
            }
        ]
    ]
}
{
    "version": "1.1",
    "sid": "PolicyService",
    "svcVersion": "0.1",
    "oid": "addPermissions",
    "params": [
        "Mashzone",
        "type.entity.uiObject",
        "accessFeedEditor",
        [
            {
                "principalId": "userID",
                "principalTypeId": "User"
            }
        ]
    ]
}
```

3. Enter the following code blocks if you want to assign the permissions to a specific user group. Enter the relevant group name for the `groupID` parameter.

```
{
    "version": "1.1",
    "sid": "PolicyService",
    "svcVersion": "0.1",
    "oid": "addPermissions",
    "params": [
        "Mashzone",
        "type.entity.uiObject",
        "accessDashboardEditor",
        [
            {
                "principalId": "groupID",
                "principalTypeId": "Group"
            }
        ]
    ]
}
{
    "version": "1.1",
    "sid": "PolicyService",
    "svcVersion": "0.1",
    "oid": "addPermissions",
    "params": [
        "Mashzone",
        "type.entity.uiObject",
        "accessFeedEditor",
        [
            {
                "principalId": "groupID",
```

```
                "principalTypeId": "Group"
            }
        ]
    ]
}
```

4. Click **Run**.

The permissions to create and edit dashboards and data feeds are assigned to the relevant user or user group.

Enable dashboard and data feed creation

The creation of dashboards and data feeds is restricted in MashZone NextGen by default. You can enable a user to create dashboards and data feeds by assigning the user to the **Presto_Developer** group.

If an existing MashZone NextGen repository is used (e.g., after a migration to a newer MashZone NextGen version) the permissions might be missing in the **Presto_Developer** group. You can add the required permissions to the repository.

Procedure

1. Open the PolicyEngineConfiguration.xml file located in the following directory using a text editor.

   <MashZone NextGen installation>\apache-tomcat\webapps\mashzone\WEB-INF\classes\

2. Add the following code block to the **Presto_Developer_PermissionSet** and save the file.

```
<Permission resourceName="Mashzone" resourceType="type.entity.uiObject">
    <Action name="accessDashboardEditor"/>
    <Action name="accessFeedEditor"/>
</Permission>
```

3. Open the API Console.
4. Enter the following code block.

```
{
    "sid": "PolicyService",
    "oid": "reloadPermissionSets",
    "version": "1.1",
    "params": {}
}
```

5. Click **Run**.

The permissions are added to the MashZone NextGen repository.

Install MashZone NextGen as a Windows service

You can install MashZone NextGen as a Windows service by running the mashzonenextgen-service.bat file.

The file is located in the <*MashZone NextGen installation*> \apache-tomcat\bin directory.

Procedure

1. Run mashzonenextgen-service.bat -install to install MashZone NextGen as a Windows service.
2. Run mashzonenextgen-service.bat -remove to uninstall MashZone NextGen as Windows service.

The Windows service **Software AG MashZone NextGen <version>** is installed or uninstalled.

Command Central plug-in

Using the Software AG Command Central web user interface and command line interface you can manage some MashZone NextGen functionalities in your Software AG products environment.

The Command Central is a tool to administer hundreds of managed product installations in your IT landscape from a central location. Command Central is built on top of Software AG Common Platform and product-specific features are in the form of plug-ins. MashZone NextGen provides an own Command Central Platform Manager plug-in. The MashZone NextGen plug-in is available as a separate item in the SAG Installer product tree. It is automatically selected when installing the MashZone NextGen product.

Detailed information on how to use the Software AG Command Central can be found in the Command Central User Guide.

Instance Overview

- The **Instances** tab provides information about the run-time status and installation details.
- By clicking MashZone NextGen on the **Instances** tab you can open the **Overview** page of the MashZone NextGen instance. The overview tab contains information about the runtime status and allows triggering some life-cycle actions like starting and stopping the instance.
- By clicking on the status icon or the link below you can start or stop the MashZone NextGen instance.
- By clicking the flag icon you can open the alerts. You can also clear the alerts by clicking on the number in the circle.

The KPIs shown are available only if the server is configured for remote JMX connections, see "Remote JMX connection" on page 52 for a configuration example. The following table gives an overview of the supported KPIs.

KPI	Description	Max value	Marginal value	Critical value
Running Calculation	The number of currently running feed calculations	10	4	8
Heap Memory	The actually used heap memory of the JVM	Configured max. value (parameter "wrapper.java. maxmemory" in wrapper.conf)	80% of max	90% of max
Calc. Queue Size	The number of waiting feed calculations in the queue	20	8	16

Instance Configuration

The instance configuration supports four different configuration types:

- General properties contained files "presto.config" and "mashzone.properties"
- Java service wraper properties contained in file "wrapper.conf" resp. "custom_wrapper.conf"
- Java system properties contained in file "wrapper.conf" resp. "custom_wrapper.conf"
- Memory configuration contained in file "wrapper.conf" resp. "custom_wrapper.conf"

The following table gives an overview of the supported configuration types, their location and which configuration aspects they cover.

Configuration type	Configuration instance	Location	Covered aspects
General Properties	MashZone NextGen properties	<TOMCAT_ROOT>/webapps/ WEB-INF/mashzone.properties	■ Initial resource folder ■ Import/Export folder ■ JDBC drivers folder ■ Feed calculation settings ■ Feed related and general cache settings ■ RTBS endpoit URL ■ PPM connection timeout
General Properties	Presto Config	<TOMCAT_ROOT>/webapps/ WEB-INF/classes/presto.config	■ UI mode "MashZone" or "Classic" ■ SAML2 configuration ■ JWT configuration ■ Landing page welcome text ■ Parts of the SSL configuration (selfsigned, anyhosts) ■ Parameters to control DAO/DB code generation ■ Database DAO service registration options
Memory	-	<TOMCAT_ROOT>/conf/ wrapper.conf <TOMCAT_ROOT>/conf/ custom_wrapper.conf	■ JVM initial heap size ■ JVM maximum heap size
Jaca Service Wrapper	-	<TOMCAT_ROOT>/conf/ wrapper.conf	■ Java home and Java command locations

Configuration type	Configuration instance	Location	Covered aspects
		<TOMCAT_ROOT>/conf/ custom_wrapper.conf	■ Java Main class to be used ■ Java classpath ■ Wrapper log settings ■ Wrapper Display Name ■ Windows service settings ■ Wrapper files locations ■ ...and some more settings of lower importance
Java System Properties	-	<TOMCAT_ROOT>/conf/ wrapper.conf <TOMCAT_ROOT>/conf/ custom_wrapper.conf	■ Tomcat location settings ■ File encoding ■ Some RAQL settings ■ Presto classic resource folder ■ Terracotta license key and config timeout ■ Derby locks settings ■ Some file locations concerning event services ■ Optional: Remote JMX settings ■ Optional: Remote debugging settings

<TOMCAT_ROOT> usually is <INSTALL_ROOT>/MashZoneNG/apache-tomcat.

The latter three configuration types are handled by the Tanuki Service Wrapper. It supports a "cascaded" configuration. That means, that only the default configuration is contained in file "wrapper.conf". Parameters that were added or manipulated are stored in file "custom_wrapper.conf" which is included by "wrapper.conf". If parameters are contained in both files, the ones in "custom_wrapper.conf" overwrite the original values in "wrapper.conf".

Note: Some important configuration aspects of MashZone NextGen are currently not supported by Command Central. They can be configured by using the

MashZone NextGen user interface or by editing specific configuration files. Detailed information on how to configure MashZone NextGen can be found in "MashZone NextGen Server Configuration" on page 87.

Instance Logs

The **Logs** tab provides the MashZone NextGen's log files located in the <TOMCAT_HOME>/logs directory.

Required installer modules

If you want to use the MashZone NextGen Command Central integration, you need to install the following modules from the installer's product tree.

- MashZone NextGen#Business Analytics 10.1
- Infrastructure# Command Central
- Infrastructure# Platform Manager 10.1
- Infrastructure# Platform Manager Plug-Ins#MashZone NextGen Plug-in 10.1

Remote JMX connection

In order to enable the remote JMX connection that allows monitoring the KPIs, please add the following parameters via Command Central Configuration Type "Java System Properties" and restart the server using the Command Central life-cycle actions.

```
com.sun.management.jmxremote.port=9607
com.sun.management.jmxremote.authenticate=false
com.sun.management.jmxremote.ssl=false
```

The SPM plug-in detects the JMX port automatically.

Type	Value	Description

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 Integration to allow users to use PPM 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 169 for links to instructions.
- Configuration for MashZone NextGen connections and subscriptions to Apama events. See "Event Service Configuration and Administration" on page 169 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, loggingoptions, proxy server configuration or caching configuration.
  - A wide range of environment or business-specific configurationsuch as connecting to SharePoint for the MashZone NextGen Add-On for SharePointor managing provide and category taxonomies.
- Configuration for authentication and authorization. See "MashZone NextGen Security" on page 55 for instructions.
- Other MashZone NextGen Repositories tasks.
- Administration tasks are discussed in MashZone NextGen Server Administration for:
  - Viewing and managing server logs.
  - Managing resource files.
  - Migrating to new MashZone NextGenreleases.
  - Deploying MashZone NextGeninstances and artifacts.
  - Clustering MashZone NextGen Servers.

2 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.

- **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 57.
- **Change password of target data sources**: For reasons of security, we strongly recommend to change the key that is used to encrypt or decrypt passwords of target data sources (e.g., source operators, URL aliases, JDBC configurations). The key is included in the authTokenKey file located in <*MashZone NextGen installation*> /webapps/mashzone/WEB-INF/classes/. It can be changed by using the `padmin generateKey -a AES -f authTokenKey` command that creates a new authTokenKey file. First of all we recommend to create a backup of the existing authTokenKey file and then to copy the new file to that folder. The file should only be changed with an empty repository, as already encrypted passwords can not be decrypted any longer. The same applies to exported content. The system where the content should be imported, has to use the same key to be able to decrypt the passwords.
- **User Authentication**: based on the protocols shown above. You can also allow anonymous access if needed. See "Authentication and Guest Access" on page 58 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 74 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.
- **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 158 and "Configure the Default Operations Generated for Database Mashable" on page 151 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-tomcat/webapps/mashzone/WEB-INF/classes/mz.properties

<*MashZoneNG-install*> /apache-tomcat/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 60 for more information.
- Valid credentials. See "Valid Credentials" on page 59 for more information.
- Guest access header or parameter information. See "Enabling Guest Access" on page 60 for more information.

User Authentication

MashZone NextGen is initially installed with a set of "Default User Accounts" on page 60 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 41, "Manage Users" on page 42 and "Manage User Groups" on page 43 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 69 for configuration instructions.

- A configurable Single Sign-On solution

  See "Authentication with Single Sign-On Solutions" on page 61 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 41 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 41 and "Authentication with Digital Certificates/SSL" on page 69 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 61 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 60 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* /mashzone/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.

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 77 for more information on permissions.

Username	Password	Built-in Group / Permissions	Description
`Administrator`	`manage`	`Presto_Administrator`	A MashZone NextGen administrator.

Username	Password	Built-in Group / Permissions	Description
dev	devdev	Presto_Developer	A developer.
power	powerpower	Presto_PowerUser	A domain expert or power user.
user	useruser	Presto_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 62 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.

- 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.

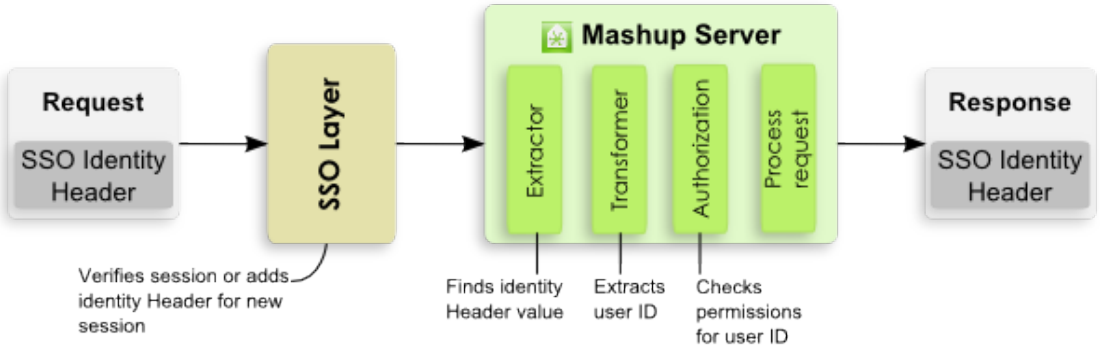
See "Configuration for the CAS SSO Solution" on page 65 for instructions.

- MashZone NextGen provides the integration under My webMethods in a SSO scenario by SAML (Security Assertion Markup Language).

  See "SSO integration in My webMethods" on page 68 for details.

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 68 for details.

1. If needed, configure the MashZone NextGen User Repository. See "Use the Default MashZone NextGen User Repository" on page 41 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* /mashzone/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:

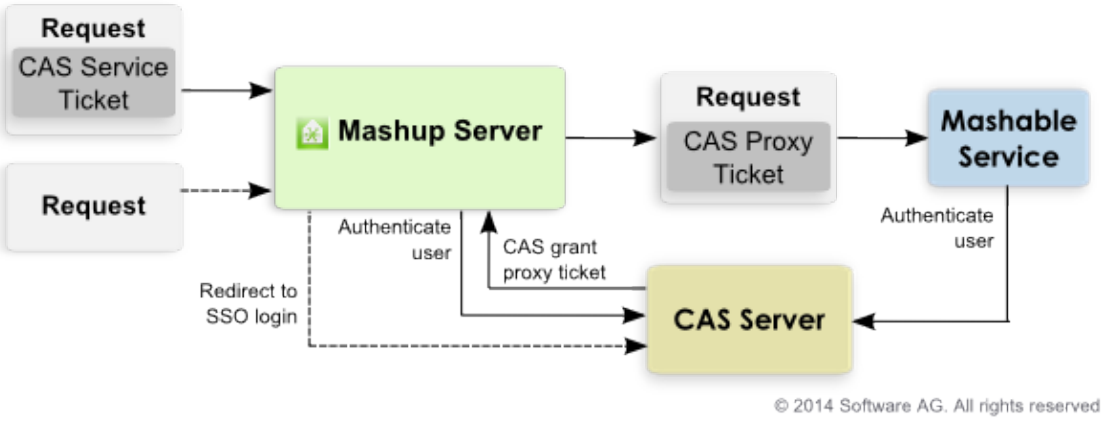
```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    <property name="principalExtractor">
      <bean
>
        <property name="httpHeaderName" value="SM_USER"/>
      </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 17 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 110 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 107 for more information.

2. Open applicationContext-security.xml in any text or XML editor.

   This file is located in the *web-apps-home* /mashzone/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* /mashzone/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* /mashzone/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 17 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* /mashzone/WEB-INF/classes folder.
   - The *web-apps-home* /mashzone/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 280 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* /mashzone/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
   - *web-apps-home* /mashzone/WEB-INF/lib. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.

SSO integration in My webMethods

You can integrate MashZone NextGen under My webMethods in an SSO scenario by SAML (Security Assertion Markup Language).

MashZone NextGen can accept SAML tokens for authentication in a SSO environment. Specifically, My webMethods can act as an Identity Provider (IdP).

MashZone NextGen verifies the signature used to sign the SAML assertion is trusted by looking the comparing the signature to the platform_truststore.jks file. This file is a Java Keystore file, and can be managed using the Java "keytool" command. If the certificate used to sign the SAML assertion is not present in the platform_truststore.jks file, the

assertion is rejected. The platform_truststore.jks file is configurable in *SAG_HOME* / MashZoneNG/apache-tomcat/webapps/mashzone/WEB-INF/classes/presto.config.

Information on the Java "keytool" command can be found in the Java documentation: "http://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html" on page

1. Within the presto.config file, the `saml.truststore.file` parameter contains the full path to the file. The default configuration uses the *SAG_HOME* /common/conf/ platform_truststore.jks file. By default, the file contains the certificate used to sign My webMethods SAML assertions. No further configuration is needed in the My webMethods SAML case.
2. Within the presto.config file, the `saml.truststore.passwd` parameter contains the keystore password. The default configuration uses the password for the *SAG_HOME* /common/conf/platform_truststore.jks file. The default password is **manage**.
3. To accept SAML assertions signed by a third party, the signing certificate must be either imported as a "trusted certificate" to the currently configured platform_truststore.jks file, or the presto.config file must be altered to point to a different keystore file, where this signing certificate is already imported as a "trusted certificate".

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 105 for instructions.
2. Using any text or XML editor, edit the applicationContext-security.xml file in the *web-apps-home* /mashzone/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 70 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* /mashzone/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* /mashzone/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,
      sharepointSSOFilter,soapRequestAuthenticationFilter, basicProcessingFilter,
      anonymousProcessingFilter, exceptionTranslationFilter
      /**/edge/api/mashsoap/**=basicReqFlowSupportFilter,  sessionContextIntegrationFilter,
      sharepointSSOFilter, soapRequestAuthenticationFilter, basicProcessingFilter,
      anonymousProcessingFilter, exceptionTranslationFilter
      /**/api/soap/**=basicReqFlowSupportFilter,  sessionContextIntegrationFilter,
      sharepointSSOFilter, wsSecurityProcessingFilter, basicProcessingFilter,
      anonymousProcessingFilter, exceptionTranslationFilter
     /**/api/rest/**=restReqFlowSupportFilter,  sessionContextIntegrationFilter,
     sharepointSSOFilter, restLogoutFilter, restLoginProcessingFilter, x509ProcessingFiler,
     basicProcessingFilter,  anonymousProcessingFilter, sessionTimeoutDetectionFilter,
     exceptionTranslationFilter
     /**/emml/debug=basicReqFlowSupportFilter, sessionContextIntegrationFilter,
     ssoProcessingFilter,restLogoutFilter, restLoginProcessingFilter, basicProcessingFilter,
     anonymousProcessingFilter, exceptionTranslationFilter
     /**/api*= jumpReqFlowSupportFilter, sessionContextIntegrationFilter,ssoProcessingFilter,
     jumpLogoutFilter, jumpLoginProcessingFilter, x509ProcessingFiler, basicProcessingFilter,
     anonymousProcessingFilter, sessionTimeoutDetectionFilter, exceptionTranslationFilter,
     filterInvocationInterceptor,sessionTimeoutDetectionSupportFilter
      ...
```

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* /mashzone/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* /mashzone/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* /mashzone/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* /mashzone/WEB-INF/classes folder.
   - The *web-apps-home* /mashzone/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 280 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* /mashzone/WEB-INF/classes. This is the default location, but is not recommended as it complicates MashZone NextGen Server deployments.
   - *web-apps-home* /mashzone/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* /mashzone/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.

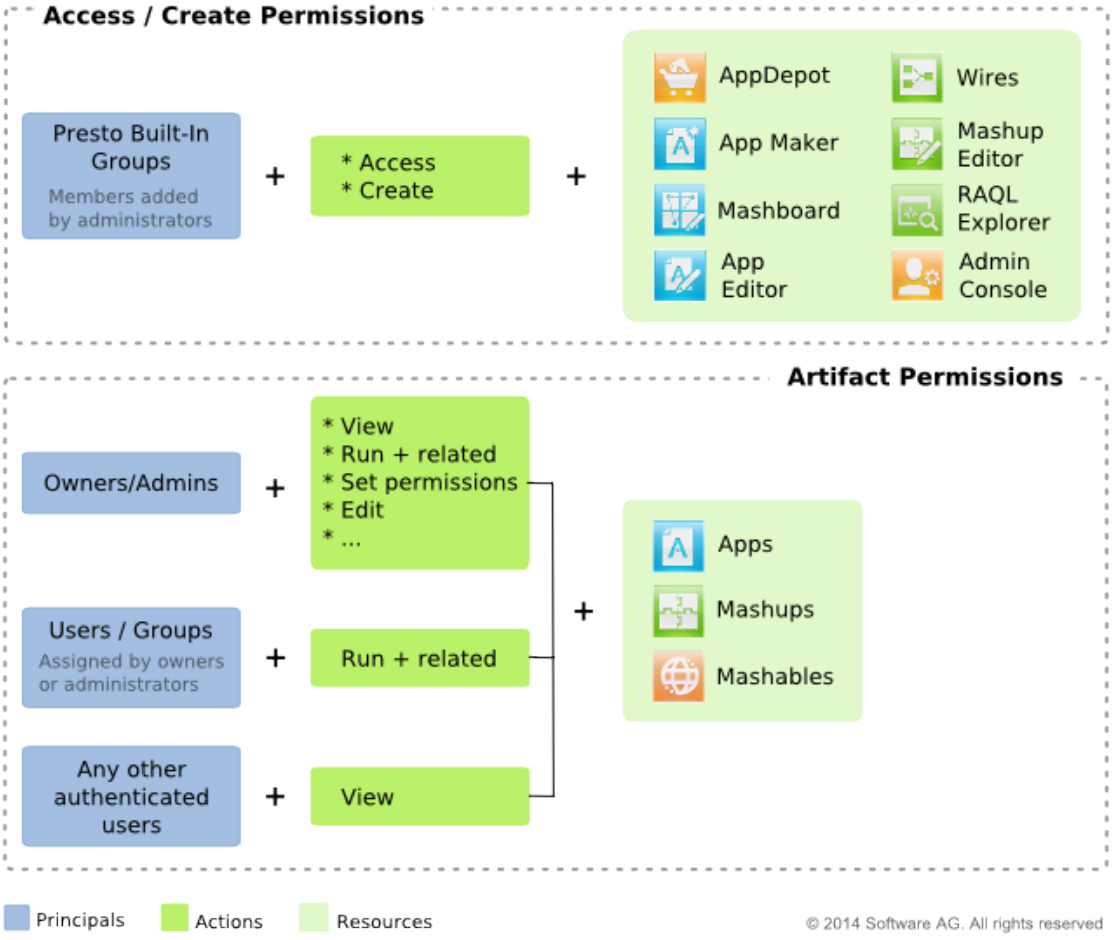
Authorization Policies and Permissions

Authorization policies determine the actions that users can perform with the mashables, mashups and apps that governs. Policies also determine user access to the features and tools in the and the 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 80 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 77 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 76 for instructions.

- **Owner/Admin Permissions**: users automatically obtain owner permissions when they create artifacts. Administrator permissions are defined when you assign users to the `Presto_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 "Automatically Grant Run Permissions to Users and Groups" on page 79. 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 58 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 77 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 79 for information.

Grant User Access to MashZone NextGen with Built-in Groups

All users in the MashZone NextGen User Repository automatically belong to the `Presto_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 `Presto_PowerUser`, `Presto_Developer` or `Presto_Administrator` groups.

See the "Built-In MashZone NextGen User Groups and Permissions" on page 77 topic for information on the specific access policies for these groups. Or use the "Default User Accounts" on page 60 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 42 for instructions.
  2. Assign users to the appropriate built-in groups. See "Edit, Grant Permissions and other User Management Tasks" on page 43 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 44 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 `Presto_Administrator`, `Presto_Developer` and `Presto_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 77 and "Artifact Permissions for Users with Run Permissions" on page 78.

Access Policies Using MashZone NextGen Built-In Groups

- *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 `Presto_Guest` built-in group. See "Enabling Guest Access" on page 60 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 `Presto_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 `Presto_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 dashboards, data feeds, 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.

  Users of the `Presto_Developer` group have the permission to create and edit dashboards and data feeds.

- **Administrators** = users in the `Presto_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. 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**.

Protect RTBS webservice access

You can use your own keystore and truststore to protect RTBS webservice of unauthorized access.

After the installation, MashZone NextGen uses a default keystore and truststore. For security reason we recommend to change that configuration for production environments. Please make sure that the truststore, referenced by RTBS, contains the appropriate certificate for the key, referenced by MashZone NextGen. RTBS is only available, with a valid configuration.

If multiple MashZone NextGen nodes are used in a clustered scenario, it is recommended to use the same key for all MashZone NextGen instances.

The default keystore and truststore are located in the **common** and **conf** folders of the MashZone NextGen installation.

For authentication MashZone NextGen webapp sends an HTTP header "Authorization" with "Bearer [JWT]" as value.

Procedure

1. Edit the following parameters, used by MashZone NextGen, to use your own keystore file to generate the JWT required for authentication.
   - jwt.keystore.file
   - jwt.keystore.passwd
   - jwt.keystore.alias

   The parameters are contained in the **presto.config** file in the following directory.

   <*MashzoneNG_install*> \apache-tomcat\webapps\mashzone\WEB-INF\classes\
2. Edit the following parameters, used by RTBS, to use your own truststore file to verify the JWT.
   - truststore.file
   - truststore.passwd

   The parameters are contained in the **rtbs.config** file in the following directory.

   <*MashzoneNG_install*> \apache-tomcat\webapps\rtbs\WEB-INF\classes\

Anti-Clickjacking prevention when using iFrame

For security reason we recommend to configure your iFrame setting to protect your MashZone NextGen installation against clickjacking attacks.

Clickjacking is a vulnerability where an attacker creates a page that uses iFrame to render another page, then creates invisible controls on top of the rendered page that may be able to sniff user input.

General information on the clickjacking attack vector can be found on https://www.owasp.org/index.php/Clickjacking.

MashZone NextGen offers two ways to prevent successful clickjacking attacks. In order to allow iFrame on trusted sites, MashZone NextGen uses X-Frame-Options providing the **ALLOW-FROM** value. Using this, a website A can configure the header to carry the top level URI of a website B which is allowed to iframe website A. A second way to prevent clickjacking attacks is using the Content-Security-Policy that is supported by most web browsers.

Details on how to use iFrame with MashZone NextGen can be found in"Embedding MashZone NextGen in external system environments" on page 101.

MashZone NextGen HTTP header security filter

MashZone NextGen provides a specific HTTP header security filter included in the **web.xml** file. By default, this filter always sends the X-Frame-Option: **SAMEORIGIN**, that can be configured to send **ALLOW-FROM** to any number of trusted websites. This HTTP response header instructs the browser to refuse to render any content from MashZone NextGen in an iFrame, unless the iFrame is within MashZone NextGen itself.

HttpHeaderSecurityFilter

Following the commented configuration in the **web.xml** file.

```
<filter>
    <filter-name>HTTP Header Security Filter</filter-name>
    <filter-class>
    com.jackbe.jbp.sas.security.ui.http.HttpHeaderSecurityFilter
    </filter-class>
    <!-- Init Param: antiClickJackingEnabled
         Should the anti click-jacking header (X-Frame-Options)
         be set on the response.
         Valid options: true or false
         When true, X-Frame-Options will always contain "SAMEORIGIN".
         This instructs browsers to disallow iframing
         of MzNG content outside of the MzNG application itself.
         If false, X-Frame-Options will
         not be sent at all, which completely disables clickjacking protection
         allows any site to iframe MzNG)
         Note: X-Frame-Options is superseded by Content-Security-Policy.
         https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
    -->
    <init-param>
        <param-name>antiClickJackingEnabled</param-name>
        <param-value>true</param-value>
    </init-param>
    <!-- Init Param: antiClickJackingUris
         List of comma separated Uris for sites allowed to iframe content in MzNG.
         To allow external sites to iframe MzNG content, uncomment this init param,
         and add the site uri to the list.
         Also configure the 'Content Security Policy' filter below.
         If the request to MzNG contains a referer value matching the scheme,
         host and port
         of one of the Uris in the list, the X-Frame-Options header will send
         "ALLOW-FROM uri'.  This allows the browser to render the iframe.
         If there is no match (or the list is empty) X-Frame-Options will send
         "SAMEORIGIN" and the browser will refuse to render the iframe
         Any site added to this list should also be added to
         'Content Security Policy' header.
    <init-param>
        <param-name>antiClickJackingUris</param-name>
        <param-value>http://some-server.com</param-value>
    </init-param>
        -->
    <!-- Init param: hstsEnabled
         Enable HTTP Strict Transport Security (HSTS) header
         (Strict-Transport-Security) to be set on the response for
         secure requests -->
    <init-param>
        <param-name>hstsEnabled</param-name>
```

```
        <param-value>true</param-value>
    </init-param>
    <!-- Init Param: hstsMaxAgeSeconds
         The max age value that should be used in the HSTS header.
         Negative values will be
         treated as zero. If not specified, the default value of 0 will be used.
     -->
    <init-param>
        <param-name>hstsMaxAgeSeconds</param-name>
        <param-value>604800</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>HTTP Header Security Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

The **antiClickJackingUris** parameters can take a list of comma separated URIs. The parameter is commented out by default. Any request for a MashZone NextGen resource containing a "Referer" header field matching the scheme, host and port of a URI in the **antiClickJackingUris** parameter will result in a response containing the X-Frame-Options response header with the appropriate **ALLOW-FROM** value. If there is no match, then the X-Frame-Options will carry the **SAMEORIGIN** value.

Example

The website http://website-a.com is configured as trusted, and therefore it is listed in the **antiClickJackingUris** parameter, and contains a page that uses iFrame to embed a MashZone NextGen dashboard. When a user visits this page on website-a.com, the browser will attempt to fetch the iFramed dashboard from MashZone NextGen. The request generated by the browser will carry the HTTP request header "Referer" containing the full URI to the page containing the iFrame. MashZone NextGen will match the "Referer" URI with the trusted URI from **antiClickJackingUris** parameter, and recognize that the website is trusted. As a result, the response will carry the HTTP response header "X-Frame-Options: ALLOW-FROM http://website-a.com ". The browser will then allow the iFrame to render.

MashZone NextGen Content Security Policy

Most modern browsers such as Microsoft Edge, Chrome, Firefox and Safari check for the newer Content-Security-Policy HTTP header instead of X-Frame-Options. Within the MashZone NextGen**web.xml** file is a second HTTP filter class that sends the HTTP Header **Content-Security-Policy**. This filter is configured by default to send the value **frame-ancestors 'self'** which is equivalent to **SAMEORIGIN** in that it instructs the browser to only allow iFrame if the iFrame is already in the originating website.

Note: The Content-Security-Policy is not supported by Microsoft Internet Explorer.

ContentSecurityPolicy

```
<filter>
    <!--
        Allows setting of HTTP header Content-Security-Policy
        http://www.w3.org/TR/CSP2/
        To prevent clickjacking attacks default is "frame-ancestors 'self'"
        which disallows external iframing of MzNG content.
```

```
        To allow additional websites to iframe MzNG content,
        add the site Uri after 'self'.
        For example:
        "frame-ancestors http://*.example.com/ 'self'"
        https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/
        Content-Security-Policy/frame-ancestors
     -->
    <filter-name>Content Security Policy</filter-name>
    <filter-class>com.jackbe.jbp.sas.security.ui.http.ContentSecurityPolicyFilter</
     filter-class>
    <init-param>
        <param-name>policy</param-name>
        <param-value>frame-ancestors 'self'</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>Content Security Policy</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Adding a trusted site to allow iFrame

The default settings do not allow external sites to iframe internal MashZone NextGen assets such as dashboards, apps, etc. Specifically, "X-Frame-Options: SAMEORIGIN" and "Content-Security-Policy: frame-ancestors 'self'" are set, which instructs the browser to disallow rendering MashZone NextGen content in any external iFrame. Via configuration and re-start, we can relax this restriction.

1. Open the **web.xml** file in a text editor. The file is located in <*MashZone NextGen installation*> /MashZoneNG/apache-tomcat/webapps/[presto|mashzone]/WEB-INF/.
2. Find the **<filter>** entry of the HTTP Header Security Filter and uncomment the **antiClickJackingUris** parameter.
3. Replace the sample URI ' http://some-server' with the URI of the website allowed to iframe MashZone NextGen content.
4. Find the **<filter>** entry for Content-Security-Policy. Insert the URI of the website allowed to iframe MashZone NextGen content into the **policy** parameter, between **frame-ancestors** and **'self'**

   Example:

   ```
   <init-param>
         <param-name>policy</param-name>
         <param-value>frame-ancestors http://*.eur.ad.sag:* 'self'</param-value>
   </init-param>
   ```

Adding multiple trusted sites to allow iFrame

To allow more than one website, perform the steps as shown in **Adding a trusted site to allow iFrame**.

1. In the **HTTP Header Security** filter, add a comma separated list of URIs as the **antiClickJackingUris** value:

   ```
   <init-param>
         <param-name>antiClickJackingUris</param-name>
         <param-value>http://website-a.com, http://website-b.com:9999
         </param-value>
   ```

```
</init-param>
```

2. In the **Content-Security-Policy** filter, add the URI to the policy parameter value, separated by a space:

```
<init-param>
      <param-name>policy</param-name>
      <param-value>frame-ancestors
       http://website-a.com http://website-b.com 'self'
      </param-value>
</init-param>
```

Content-Security-Policy using wildcards

The Content-Security-Policy allows wildcards to be used in the policy. For example, to allow any website on any port hosted in the "eur.ad.sag" domain, you can specify:

```
<init-param>
      <param-name>policy</param-name>
      <param-value>frame-ancestors http://*.eur.ad.sag:* 'self'
      </param-value>
</init-param>
```

3 MashZone NextGen Server Configuration

Basic Configuration and Logging

- "Manage Licenses for MashZone NextGen and BigMemory" on page 18
- "Support International Character Sets and Locales" on page 92
- "Change the MashZone NextGen HubTheme" on page 96
- "Configure the MashZone NextGen server with custom ports" on page 98
- "Configure the MashZone NextGen server to work with a proxy server" on page 100
- "Define a Proxy Server Whitelist for MashZone NextGen" on page 103
- "Configure MashZone NextGen for SSL and Digital Certificates" on page 105
- Configuring "MashZone NextGen Notifications" on page 116
- Configuring "MashZone NextGen Logging" on page 113 and viewing logs
- "Manage data sources and drivers" on page 145
- "Manage Categories for Mashups, Mashables and Apps" on page 153
- "Manage Providers for Mashups, Mashables and Apps" on page 153
- "Work With MashZone NextGen Attributes" on page 154

See also "MashZone NextGen Repositories" on page 229 for links to configuration for the MashZone NextGen Repository.

Mashables, Mashups and Wires Configuration

- "Manage Artifact Attributes" on page 157
- "Manage data sources and drivers" on page 145
- "Configure the Default Operations Generated for Database Mashable" on page 151
- "Disable Mashup Features" on page 158
- "Configure Mashable HTTP Request Timeouts" on page 160
- "Configure HTTP Response Header Forwarding" on page 159
- "Enable or Disable the Snapshot Feature" on page 160
- "Set Web Feed Normalization" on page 161
- "Handle SOAP Encoding Errors for WSDL Services" on page 161
- "Add XML Schemas to the Wires Mapper Block" on page 162

Add-Ons, Performance, Deployment and Miscellaneous

- "BigMemory for Caching, Connections and MashZone NextGen Analytics" on page 124

- "Configure MashZone NextGen Connections to SharePoint" on page 118
- "Memory Configuration for the MashZone NextGen Server" on page 90
- "Deploying MashZone NextGen Instances, Clusters or Artifacts" on page 247
- See also "MashZone NextGen Security" on page 55 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 90
■ 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 91

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 2G as the maximum heap size. The Event Service that is deployed with MashZone NextGen also uses this heap space.

Note: If you are using MashZone NextGen 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.

How much memory to allocate to MashZone NextGen depends on available memory, requirement for the Event Service and what other applications may run on this computer.

To update memory configuration

1. In a text editor of your choice, open the application server configuration file *MashZoneNG-install* /apache-tomcat/conf/wrapper.conf
2. Change any of these Java memory options:

wrapper.java.initmemory	Default = `512`
wrapper.java.maxmemory	Default = `2048`

3. Save your changes and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 17 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 and BigMemory" on page 18 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 Event Service that is 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.

To update memory configuration:

1. In a text editor of your choice, open the application server configuration file *MashZoneNG-install* /apache-tomcat/conf/wrapper.conf.
2. Change or add either of these memory options used with BigMemory:

wrapper.java.additional.<n+1> =-Dpresto.bm.maxOffHeap	Default = `1G` Where n is the number of last additional Java parameter. 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.
wrapper.java.additional.<n+2> =-XX:MaxDirectMemorySize	Default = 1500M Where n is the number of last additional Java parameter. 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:

wrapper.java.initmemory	Default = `512M`
wrapper.java.maxmemory	Default = `2G`

See the Java Tuning White Paper for more information and suggestions.

4. Save your changes and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 17 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 93.

- *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 94.

- *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 95 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 95 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 repository to Microsoft SQL Server
  - Move the MashZone NextGen repository to MySQL
  - Move the MashZone NextGen repository to Oracle
  - Move the MashZone NextGen repository 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 usesto match the timezone for the MashZone NextGen Server.See "Synchronize the MashZone NextGen Repository and MashZone NextGen Server Time Zones" on page 232for instructions.
- Configure the display timezone in MashZone NextGen Hubas an offset from UTC. Note that this does not affect the actual timezonerecorded in the MashZone NextGen Repository.

  This offset is defined in the `repositoryTimezoneOffset` property in *web-apps-home* /mashzone/hub/config.jsfile. This property is undefined by default.

  Edit this JSON property, setting the number of minutes as a UTCoffset. For example, `300` sets this to Eastern Standard Timewhile `-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-tomcat/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`.

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.

You can also set the default locale for the JVM for MashZone NextGen. See "Message Log and Default Locales" on page 95 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 file <MashZoneNG-installation>/apache-tomcat/conf/wrapper.conf.
2. Add the lines just below the line which sets the last additional Java parameter:

   wrapper.java.additional.<n+1>=-Duser.county=country-code

   wrapper.java.additional.<n+2>=-Duser.language=language-code

   Where n is the number of last additional Java parameter.

   For example:

```
wrapper.java.additional.20=-Duser.county=CA
wrapper.java.additional.21=-Duser.language=fr
```

3. Save your changes to the file.
4. Restart the MashZone NextGen Server.See "Start and Stop the MashZone NextGen Server" on page 17 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.

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.
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 97 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 NextGen. Editing the style templates enables you to customize the look and feel of your dashboards and the dashboard editor, e.g., colors schemes, fonts, brand logo, or background colors.

Note: If you change the style template at the application level your settings are also applied in the MashZone NextGen Feed Editor.

The application style template file application.less is located in the following folder on the MashZone NextGen server.

<*MashZone NextGen installation*> \apache-tomcat\webapps\mashzone\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-tomcat\webapps\mashzone\hub\dashboard \assets\custom-look-and-feel\dashboard

The style templates provide basic variables that are used as default variables 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 for the dashboard name label, which will be red.

All application and dashboard variables available 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.
   a. Open a dashboard in the MashZone NextGen Dashboard Editor.
   b. Click **Manage** > **Change style template** in the dashboard main menu.
   c. To reload the application style template click **Activate**.
   d. To reload the dashboard style template select the relevant dashboard style template in the drop-down menu and click **Update**.
5. Click **OK**.

The changed style template has been reloaded into the MashZone NextGen Dashboard component and the style templates are applied 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 Tomcat.

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 Tomcat, 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-tomcat/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 17 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 Repository is 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 17 for instructions.

Tomcat 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-tomcat/conf/server.xml	`<Server>` to set the command port
	`<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 103 for more information.

To configure a proxy server

1. Start the MashZone NextGen Server.

   See "Start the MashZone NextGen Server" on page 17 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 103 for instructions.
9. Click **Save proxy settings**.

Embedding MashZone NextGen in external system environments

You can use MashZone NextGen as a component in external products, e.g., webMethods Business Console. As embedded component MashZone NextGen is enabled to send data via outbound API (Post data) to the embedding system and receive data via inbound API (URL selection) from the embedding system.

Configure MashZone NextGen server to work with iFrame

By default, MashZone NextGen can be embedded using HTML inline frames (iFrame) if the MashZone NextGen server and the server of the embedding system use the same protocol, same host and same port.

To embed MashZone NextGen within another HTML document the iFrame source points to the MashZone NextGen dashboard as shown in the following example.

```
<iframe id="embedded-mzng-dashboard"
```

```
        width="600px" height="600px"
        src="http://mzngServerHost:mzngServerPort/mashzone/hub/
             dashboard/dashboard.jsp?mzngDashboardGUID">
    <p>Your browser does not support iframes.</p>
</iframe>
```

If the embedding system is running on a different host or uses a different protocol or port, the MashZone NextGen server must be configured as follows. The MashZone NextGen server configuration file **applicationContext-security-filters.xml** needs to be configured by adding filters for **X-Frame-Options** and content security policies.

The **applicationContext-security-filters.xml** server configuration file is located in following directory. <*MashZone NextGen-install*> /apache-tomcat/webapps/mashzone/WEB-INF/classes.

Procedure

1. Open the **applicationContext-security-filters.xml** configuration file in a text editor of your choice.
2. Adapt the security settings as follows and exchange the string "http://otherServerHost:otherServerPort" with the system origin MashZone NextGen is to be embedded in.

```
<beans:beans
    xmlns="http://www.springframework.org/schema/security"...>
    ...
    <http pattern="/hub/(login|reset_password)\.html.*" security="none"
     request-matcher="regex"/>
    <http pattern="/help/.*" security="none" request-matcher="regex"/>
    <http pattern="/**/*.jsp" use-expressions="false"
          authentication-manager-ref="authenticationManager"
          entry-point-ref="mzngAuthenticationEntryPoint">
          <anonymous enabled="false"/>
          <headers>
            <!--frame-options policy="SAMEORIGIN"/-->
            <frame-options policy="ALLOW-FROM" strategy="static"
            value="http://otherServerHost:otherServerPort" />
            <!--content-security-policy policy-directives="frame-ancestors
            'self'"/-->
            <content-security-policy policy-directives="frame-ancestors 'self'
            http://otherServerHost:otherServerPort"/>
          </headers>
        <csrf token-repository-ref="csrfTokenRepository"
         request-matcher-ref="skipHttpAuthCsrfMatcher"/>
        <custom-filter ref="samlTokenProcessingFilter"
         after="PRE_AUTH_FILTER"/>
        <custom-filter ref="jwTokenProcessingFilter"
         before="CAS_FILTER"/>
        <custom-filter ref="credentialContainerFilter"
         before="EXCEPTION_TRANSLATION_FILTER"/>
    </http>
    <http pattern="/**/*.html" use-expressions="false"
          authentication-manager-ref="authenticationManager"
          entry-point-ref="mzngAuthenticationEntryPoint">
          <intercept-url pattern="/**/*.html"
                         access="IS_AUTHENTICATED_ANONYMOUSLY"/>
          <anonymous enabled="false"/>
          <headers>
            <!--frame-options policy="SAMEORIGIN"/-->
            <frame-options policy="ALLOW-FROM" strategy="static"
```

```
            value="http://otherServerHost:otherServerPort" />
            <!--content-security-policy policy-directives="frame-ancestors
             'self'"/-->
            <content-security-policy policy-directives="frame-ancestors 'self'
            http://otherServerHost:otherServerPort"/>
          </headers>
    </http>
    ...
</beans:beans>
```

3. Save changes.

Your changes will be applied with the next MashZone NextGen server start.

Further details on the topic **Using iFrame** can be found in the spring security documentation: https://docs.spring.io/spring-security/site/docs/current/reference/html/headers.html#headers-frame-options.

Post data

The **Post data** action creates an outbound API to pass data from MashZone NextGen dashboards to an embedding system, e.g., an external web application. The action is available for most dashboard components.

URL selection

With the **URL selection**MashZone NextGen provides an inbound API to receive data from an embedding system, e.g., an external web application. The action is available for most dashboard components.

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 104.
   - *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 104.

- *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 104.

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 104 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	MashZoı NextGen	Authentication	Security Profiles
"One-Way SSL to MashZone NextGen" on page 109.	✓			✓			
Mutual to MashZone NextGen See "Configure Mutual SSL Between Users and MashZone NextGen" on page 108.	✓	✓		✓		✓	
"One-Way SSL to Mashable Information Sources" on page 110.		✓	can be in either				
"One-Way SSL to Information Sources Using <directinvoke> in Mashups"		✓	✓				

	Certificate Store and Certificates		Store Configuration			MashZone NextGen Configuration	
	Key	Trust	Java	App Server	MashZo NextGen	Authentication	Security Profiles
on page 110.							
Mutual to Mashables See "Configure Mutual SSL Between MashZone NextGen and Mashable Information Sources" on page 109	✓	✓			✓		✓

See also "The Certificate Store and Certificates" on page 107 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 108 for more information.

  See "The Certificate Store and Certificates" on page 107 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 110 for instructions.

- Optional configuration in MashZone NextGen to use digital certificates for user authentication. See "Authentication with Digital Certificates/SSL" on page 69 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 108 for more information.

  See "The Certificate Store and Certificates" on page 107 for instructions.
- Configuration in MashZone NextGen for both the key store and trust store. See "Configure Certificate Stores in MashZone NextGen" on page 112 for instructions.
- Security Profile configuration for each mashable information source. You provide this configuration when you register the mashable.

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 107 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 110 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 107 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 110 for instructions.
  - Java. See "Update SSL Configuration for Java" on page 112 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 108 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 107 for more information.
- Configuration for the trust store in Java. See "Update SSL Configuration for Java" on page 112 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 108 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 107 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 98 for more information on this step.

   c. Enable MashZone NextGen to use secure session cookies:

      a. Open the web.xml file located in <MashZone NextGen installation>/apache-tomcat/webapps/mashzone/WEB-INF/ in a text editor.

      b. Find the `session-config/cookie-config/secure` element and change the value to `true`.

      Example

```
<session-config>
        <session-timeout>30</session-timeout>
        <!--
            Set the "secure" flag to true when using HTTPS for enhanced security
        -->
        <cookie-config>
         <secure>false</secure>
        </cookie-config>
    </session-config>
```

      Note: Once this is set to true, only HTTPS access will be allowed.

3. If needed, enable MashZone NextGen authentication to use certificates. See "Authentication with Digital Certificates/SSL" on page 69 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 application server configuration file *MashZoneNG-install* /apache-tomcat/conf/wrapper.conf in a text editor of your choice.
2. Add the following Java system properties:

   wrapper.java.additional.<n+1>=-Djavax.net.ssl.trustStore=/path/to/mashup-server/truststore

   This is the absolute path to the trust store for the MashZone NextGen Server.

wrapper.java.additional.<n+2>=-Djavax.net.ssl.trustStorePassword=truststore-password

This is only required if the MashZone NextGen Server's trust store uses a password.

Where n is the number of last additional Java parameter.

3. Save your changes to the script and restart the MashZone NextGen Server. See "Start and Stop the MashZone NextGen Server" on page 17 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 113 and "View the MashZone NextGen Server Log" on page 236 for more information.
- Audit logging tracks the invocation of each mashable or mashup. See "Turn Audit Logging On or Off" on page 115, "Purge the Audit Log" on page 238 and "View the Audit Log for a Mashable, Mashup or App" on page 237 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 113 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-tomcat/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.

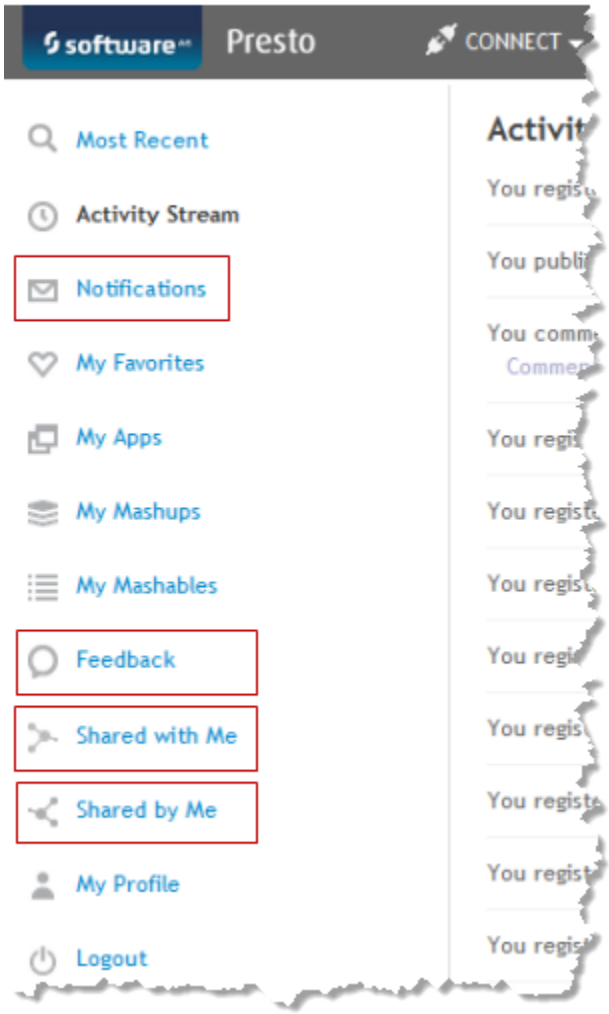
3. To turn audit logging on for a specific event, set the **Enable** option for that event.

> **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 238.

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 117 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 98 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 118 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 Console in the MashZone NextGen Hub main 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 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 119 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.

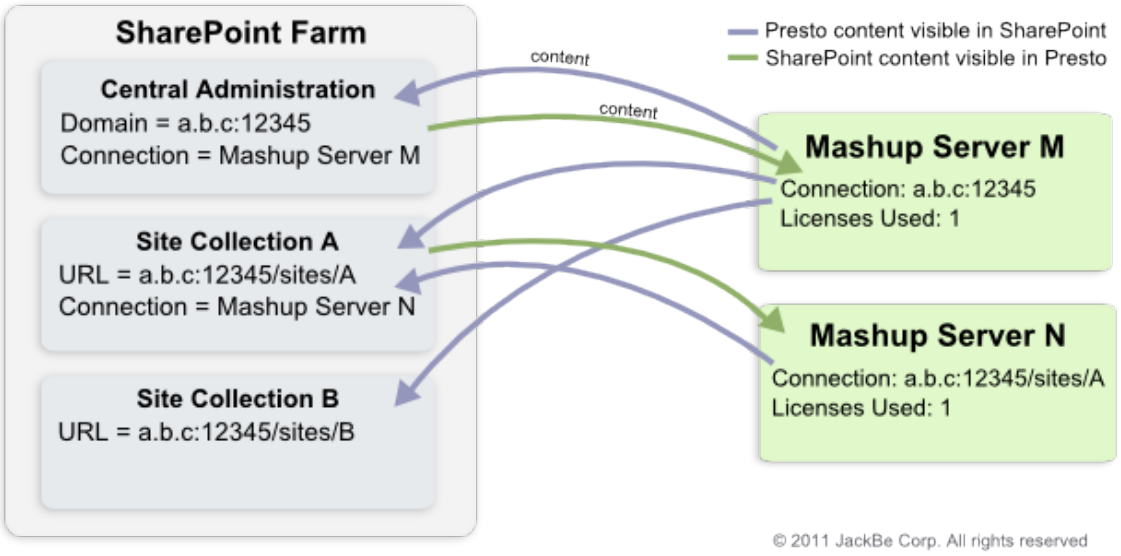
Connection Patterns Between MashZone NextGen Servers and SharePoint

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.

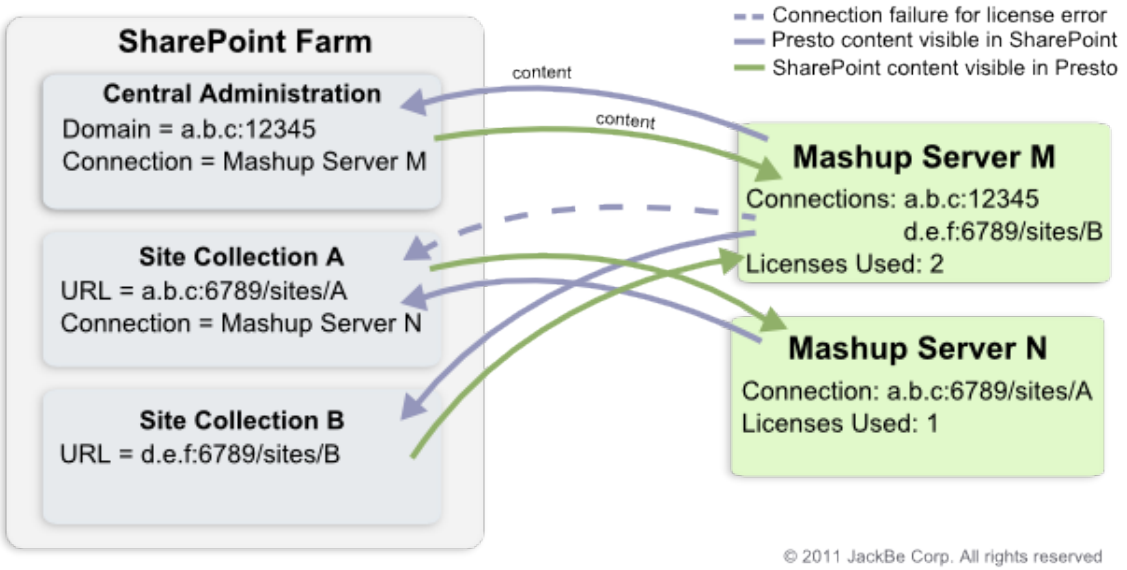
In MashZone NextGen, however, only the SharePoint Lists defined in Central Administration are visible. To make SharePoint Lists from Site Collection A and Site Collection B visible in MashZone NextGen, you would need to add separate connection configurations for each site collection in MashZone NextGen 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 119 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.

> **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 127 for instructions on configuring MashZone NextGen to work with BigMemory Servers.

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 130
- "Manage Dynamic BigMemory Stores for MashZone NextGen Analytics" on page 133.

See "Caching for the MashZone NextGen Server" on page 125 for an overview of MashZone NextGen caching. For caching configuration, see "Distributed Caching for MashZone NextGen Clusters" on page 126.

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 126 and "Response Caching" on page 126 for details.

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 126 for an overview of distributed caching for MashZone NextGen.

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 135 for instructions.

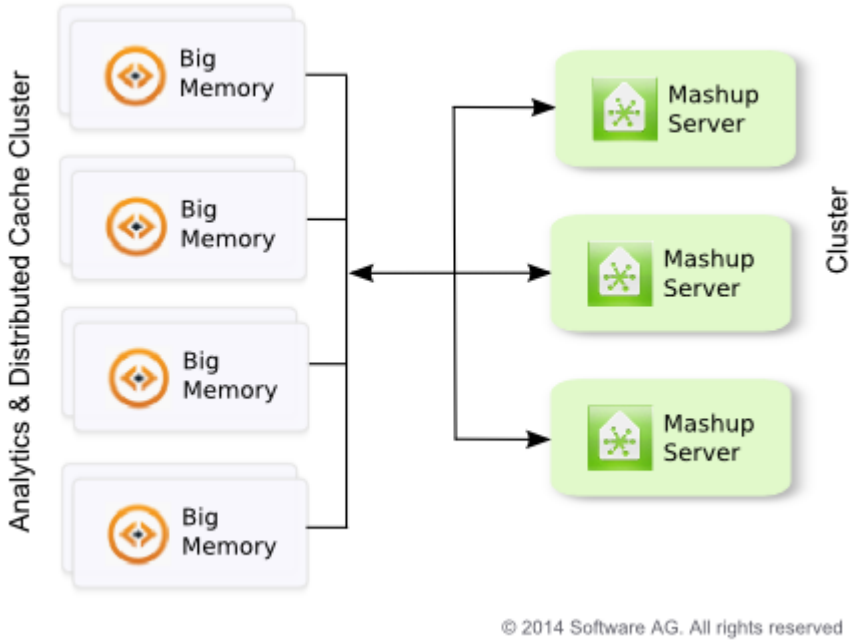
Distributed Caching for MashZone NextGen Clusters

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.

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 127 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 124 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-tomcat/conf folder.

Note: If MashZone NextGen is deployed in a cluster, you must copy this file to every cluster member.

b. Add the following Java system property to the MashZone NextGen server configuration file <*MashZone NextGen installation*> /apache-tomcat/conf/wrapper.conf:

```
wrapper.java.additional.<n+1>=-Dcom.tc.productkey.path=MashZoneNG-install/apache-tomcat/conf/terracotta.key
```

Where n is the number of last additional Java parameter.

Note: If MashZone NextGen is deployed in a cluster, you must update the server configuration files 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-tomcat/webapps/mashzone/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 280.
A MashZone NextGen Cluster With No Shared Configuration	You must update *MashZoneNG-install* /apache-tomcat/webapps/mashzone/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-tomcat/webapps/mashzone/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 280.
A MashZone NextGen Cluster With No Shared Configuration	You must update *MashZoneNG-install* /apache-tomcat/webapps/mashzone/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 133 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 90 for instructions.
10. Restart MashZone NextGen. See "Start and Stop the MashZone NextGen Server" on page 17 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 133 for more information.

See "Declare a New In-Memory Store" on page 130, "Modify a Declared In-Memory Store" on page 133 and "View Details for Declared In-Memory Stores" on page 133 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.

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-tomcat/mashzone/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 280.

3. Restart the MashZone NextGen Server. For instructions, see "Start and Stop the MashZone NextGen Server" on page 17.

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 240 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 130 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 130 for more information.

For information on the requirements for in-memory stores that act as dynamic external stores for MashZone NextGen Analytics. For instructions on adding and managing external dynamic store configuration, see "Add an External Dynamic In-Memory Store Connection" on page 134 and "Delete External Dynamic In-Memory Store Connections" on page 135.

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.

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-tomcat/mashzone/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 280.

3. Restart the MashZone NextGen Server. For instructions, see "Start and Stop the MashZone NextGen Server" on page 17.
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 distributed caching for responses. See "Configure BigMemory Servers for MashZone NextGen Caching and In-Memory Stores" on page 127 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 NextGenadministrator enables the Response Cache for the MashZone NextGen Server.

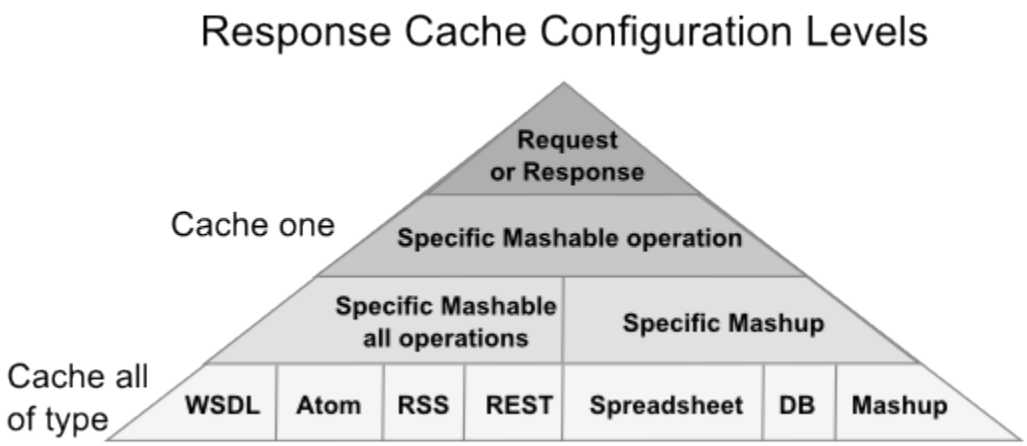
   a. Click Admin Console in the MashZone NextGen Hub main menu.

   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 139for 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 137for instructions.

   If you only need to cache responses for specific mashables or mashups, it is best to leave caching disabled for all mashable types andmashups and enable caching only for specific exceptions. See "Cache Responses for Exceptions Only" on page 138for instructions.

   You can also override cache settings for individual requests orresponses. See "Controlling Response Cache Entries Dynamically" on page 139for 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.

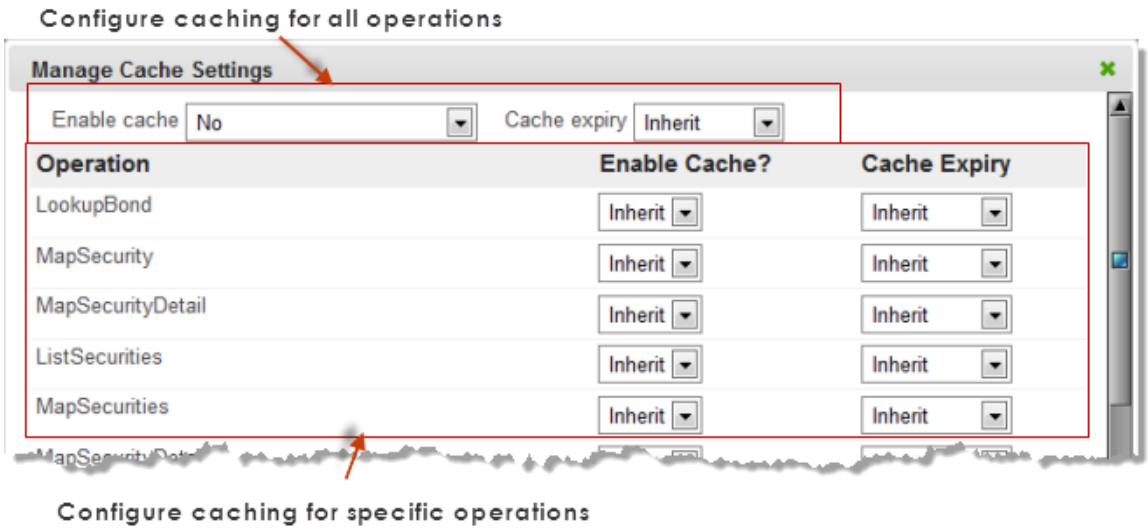
Note: Only MashZone NextGen administrators can configure response caching for all mashables of one type or all mashups. Only owners or MashZone NextGen administrators can configure response caching for a specific mashable or mashup.

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**.

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.

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.

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. Within seconds UserA adds a filter block to his mashup, connects the Yahoo Financial block and previews the result of the filter.	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. 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.	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

	Event	Cache Entry
	UserB uses this app. Three minutes later, UserC also uses this app.	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 Terracotta DB connections

You can register, edit, delete and test Terracotta DB connections. Additionally, you are able to assign ACLs to an existing Terracotta DB connection.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Server** to expand this section of the Administration menu.
4. Click **Terracotta DB**.

5. Follow the procedure of the remaining steps:
   - "Register Terracotta DB connections" on page 142
   - "Edit Terracotta DB connections" on page 142
   - "Test Terracotta DB connections" on page 143
   - "Delete Terracotta DB connections" on page 143
   - "Share Terracotta DB connections" on page 144

Register Terracotta DB connections

You can configure a connection to the Terracotta DB server. The alias is used to refer to that specific connection.

By having a connection to the Terracotta DB server configured, the dashboard developer can access all datasets available on that server. The datasets are provided in the corresponding **Terracotta DB** source operator. The alias is unique with respect to all Terracotta DB connections. Different connections can point to the same Terracotta DB server.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Server** to expand this section of the Administration menu.
4. Click **Terracotta DB**.
5. Click **Register new Terracotta DB connection**.
6. Enter a name for the Terracotta DB connection in the **Terracotta DB alias** field. The connection data is saved under this alias.
7. Enter the URI to the Terracotta DB. The format is **terracotta://hostname:port**.
8. Click **Add connection**.

The **Terracotta DB** connection is created and listed by alias name.

Click ▶ **Test connection** for the alias created to test the Terracotta DB server connection.

Edit Terracotta DB connections

You can edit the URI of an already existing connection to the Terracotta DB server. The alias is not editable.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Server** to expand this section of the Administration menu.
4. Click **Terracotta DB**.
5. Click **Edit**.
6. Enter the URI to the Terracotta DB. The format is **terracotta://hostname:port**.
7. Click **Update connection**.

Your changes are applied.

Click **Test connection** for the alias changed to test the Terracotta DB server connection.

Test Terracotta DB connections

You can test the connection to the Terracotta DB server if the connection works correctly.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Server** to expand this section of the Administration menu.
4. Click **Terracotta DB**.
5. Click **Test connection** for an alias to test the Terracotta DB server connection.

The server connection is tested and a test result is displayed.

Delete Terracotta DB connections

You can delete a Terracotta DB server connection.

> A Terracotta DB server connection deleted can not be restored. Deleting a connection affects dashboards using datasets available over that connection as data input.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Server** to expand this section of the Administration menu.

4. Click **Terracotta DB**.
5. Click **Delete**.

The Terracotta DB connection selected is deleted.

Share Terracotta DB connections

You can share Terracotta DB connections with particular users and user groups so that these have access to the Terracotta DB dataset.

Regardless of the share, users with administration privilege can access all Terracotta DB aliases.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **Server** to expand this section of the Administration menu.
3. Click **Terracotta DB**.
4. Click the **Edit URL alias permissions** icon of the Terracotta DB connection you want to share.
5. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
6. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
7. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the Terracotta DB connection is already present in the **Principals with permissions** list. This owner is non editable and cannot be removed from the list.

8. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

9. Click **Ok**.

Your changes are applied.

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.

See "Add a data source" on page 145, "Edit, test or remove data sources" on page 147 and "Add or manage JDBC drivers" on page 148 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 user menu of the program bar.
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 148 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

Procedure

1. Click **Admin Console** in the user menu of the program bar.
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 145 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.

Share data sources

You can share data sources with particular users and user groups so that these have access to *JDBC data sources*.

You have administration privileges.

Regardless of the share, users with administration privilege can access all data sources.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **JDBC Configuration** to expand this section of the Administration menu.
3. Click **Data Sources**.

4. Click the **Edit data source permissions** icon of the data source you want to share.
5. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
6. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
7. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the data source is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

8. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

9. Click **Ok**.

Your changes are applied.

Add or manage JDBC drivers

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Expand the **JDBC Configuration** tab.
4. Click **JDBC Drivers**.

   A list of existing drivers displays. This initially contains just the driver for the default MashZone NextGen Repository.

5. To remove a driver, click **Delete** on the line for that driver.

   Important: Do *not* delete the driver for the MashZone NextGen Repository.

6. 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**.

Migrate 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. The current version of MashZone NextGen MashZone NextGen supports the import of JDBC connections from MashZone legacy, Presto legacy and MashZone NextGen version 9.10.

Migrate JDBC configuration of Presto to MashZone NextGen

You can import the JDBC configuration of Presto (version 3.9 and 9.9) in MashZone NextGen.

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/mashzone/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.

Migrate JDBC connections of Presto to MashZone NextGen

You can import the JDBC connections of Presto (version 3.9 and 9.9) in MashZone NextGen.

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 the presto.config file located in <*Presto installation*> \apache-tomcat\webapps\mashzone

\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:

   ```
   migrationtool -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.

Migrate JDBC configuration of MashZone NextGen 9.10

You can import the JDBC configurations of MashZone NextGen version 9.10 in the current MashZone NextGen version.

Procedure

1. To export all existing JDBC configurations from MashZone NextGen 9.10 go to the \prestocli\bin folder of the MashZone NextGen 9.10 installation, open a dos command line and call:

   ```
   pAdmin exportDatasource -l http://localhost:8080/mashzone/esd/api -f
   JDBCConnections_backup.zip -u Administrator -w manage -j
   ```

   The created zip file contains all information about JDBC configurations of MashZone NextGen 9.10, including the related drivers.

2. Copy the JDBCConnections_backup.zip file to the prestocli\bin folder in your current MashZone NextGen installation.

3. Go to prestocli\bin folder in your current MashZone NextGen installation, open a dos command line and call:

```
pAdmin importDatasource -f JDBCConnections_backup.zip -u
Administrator -w manage -o
```

All MashZone NextGen 9.10 JDBC configurations will be imported into the current MashZone NextGen version.

Migrate JDBC connections of MashZone legacy to MashZone NextGen

You can import the JDBC connections of MashZone legacy (versions 9.5 to 9.12) in MashZone NextGen.

See the **MashZone NextGen Migration Guide** for details.

Configure the Default Operations Generated for Database Mashable

When users or administrators register simple or custom mashables for databases, generates a default set of operations for the tables, views or stored procedures in the selected database.

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 `find`*`table-name`*`ByWhereClause` operation for tables in a given database mashable.

  This option is enabled, by default.

- *Include selectTable operations* = this option determines the default availability of the `select`*`table-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 `select`*`table-name`* operation by default.

4. Change any of the options for Large Queries:

   - *Include findAllTable operations* = this option determines the default availability of the `findAll`*`table-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 `find`*`table-name`*`Where` 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 `insert`*`table-name`* operations for database mashables which insert records in database tables. This is set by default.

   - *Include update operations* = determines the default availability for `update`*`table-name`* operations for database mashables which update records in database tables. This is set by default.

   - *Include delete operations* = determines the default availability for `delete`*`table-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.

  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 156 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 155 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.
  - **In mashup scripts**.
- **Request Attributes**: can be used to refer to HTTP headers or cookies for a specific request.
- **Artifact Attributes**: are additional meta data that you provide for individual apps, mashables or mashups. See "Manage Artifact Attributes" on page 157 for more information.

Manage 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 118.
- Schemas for datasets used in RAQL queries in MashZone NextGen Analytics.

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 17 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.

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 34 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 157 for instructions. You or other artifact owners can then set the values for these attributes.

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. 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.

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 timeoutperiod for connections to be successfully made with mashable informationsources.
   - Enter a number of milliseconds in **Socket Timeout** to define the timeoutperiod 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.

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 280 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 17 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 240 for more information.

4 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 ARIS PPM.

To integrate the MashZone Server with MashZone NextGen, you must first:

- Move the MashZone Repository to a robust solution. See "Move the MashZone NextGen repository to a robust database solution" on page 19 for instructions.
- "Tune Memory/Caching for the Integrated MashZone Server" on page 164
- "Event Service Configuration and Administration" on page 169 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.

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 90 .

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 Cache Memory Settings
2. Update MashZone ThreadSize Properties.
3. Then restart the MashZone NextGen Server to apply this change. See "Start and Stop the MashZone NextGen Server" on page 17 for instructions.

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 90 for instructions.

5 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.

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*.

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, DES, or Apama event.

Use Events as Information Sources

To use events as information sources, you must create EDA, DES, or Apama event sources. Unlike other sources, only MashZone NextGen administrators can create EDA, DES 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 171. This defines the types of events that are available for subscriptions from the EDA event source.
2. For DES event source, "Identify the DES repository directory" on page 197. This defines the types of events that are available for subscriptions from the DES event source.
3. Create event sources as needed. MashZone NextGen retrieves event data from these event sources when users run the corresponding EDA, DES 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 DES Event Source" on page 197
   - Create Apama Event Sources
4. "Start or Stop an Event Source" on page 208

You can also Manage Apama Event Sources, "Manage DES Event Sources" on page 196, and Manage EDA Event Sources.

Manage EDA Event Sources

To identify, create, edit, delete, import or export *EDA Event Sources:*

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **EDA** tab.
6. Select further steps:
   - "Identify the Event Type Store directory" on page 171
   - "Create EDA Event Sources" on page 172
   - "Edit EDA Event Sources" on page 177
   - "Duplicate EDA Event Sources" on page 182
   - "Delete EDA Event Sources" on page 182
   - "Share EDA Event Sources" on page 183

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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **EDA** tab.
6. Click **Event Type Store directory**.
7. Enter the path to the local Event Type Store in the **Event Type Store directory** field.
8. Click **Save**.

Your changes are applied.

Create EDA Event Sources

"You have identified the Event Type directory." on page 171

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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **EDA** tab.
6. Click **Create EDA Event Source**.
7. Set the properties for this event source. See table "EDA Event Source properties" on page 172 below.
8. Click **Save**.

The EDA Event Source is created and listed by alias name.

Table 1. 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.
Event type	yes	Click **Refresh** to update the list of **Apama** event types. 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
Filter predicates		Enter a filter expression defining the events to be published to this event source.
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 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**.

Property	Required	Description
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
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

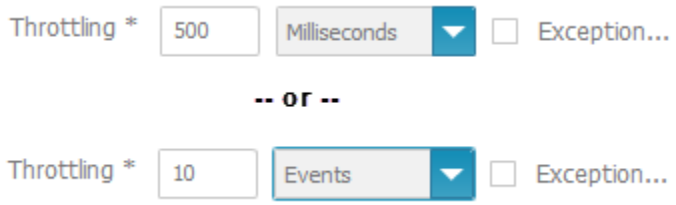
Property	Required	Description
		The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000.
Event ID attribute	yes	Available only when **Strategy** is set to `DELTA`. Select the attribute that identifies an event. The event ID and command determines 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 events are received and change the number, if needed. See "example below" on page 176.
Exception		Set this option to support a hybrid throttling strategy, typically involving both time and

Property	Required	Description
		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 176

Simple and Hybrid Throttling Strategies

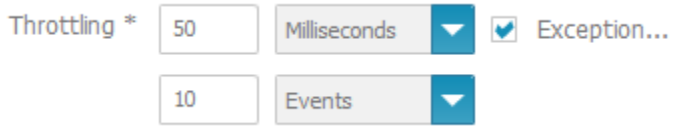
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **EDA** tab.
6. Click the **Edit** icon to configure a specific EDA connection.
7. Set the properties for this event source:

Table 2. 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.
Event type	yes	Click **Refresh** to update the list of **Apama** event types. 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
Filter predicates		Enter a filter expression defining the events to be published to this event source.
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 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**.

Property	Required	Description
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
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

Property	Required	Description
		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 events are received and change the number, if needed. See "example below" on page 176.
Exception		Set this option to support a hybrid throttling strategy, typically involving both time and

Property	Required	Description
		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 176

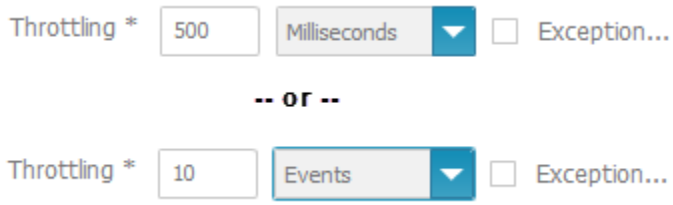
8. Click **Save**.

Your changes are applied.

Simple and Hybrid Throttling Strategies

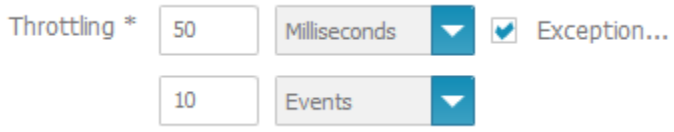
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **EDA** tab.
6. 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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **EDA** tab.
6. Click the **Delete** icon to delete a specific EDA Event Source.

The selected EDA Event Source is deleted from the list.

Share EDA Event Sources

You can share EDA Event Sources with particular users and user groups so that these have access to *EDA Event Services*.

You have administration privileges.

Regardless of the share, users with administration privilege can access all EDA Event sources.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**.
4. Open the **EDA** tab.
5. Click the **Edit event service permissions** icon of the EDA Event Service you want to share.
6. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
7. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
8. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the *EDA Event Services* is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

9. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

10. Click **Ok**.

Your changes are applied.

Manage Apama Event Sources

To create, edit, delete, import or export **Apama** Event Sources:

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **Apama** tab.
6. Select further steps:
   - "Create Apama Event Sources" on page 184
   - "Edit Apama Event Sources" on page 189
   - "Duplicate Apama Event Sources" on page 194
   - "Delete Apama Event Sources" on page 195
   - "Share Apama Event Sources" on page 195

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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.
5. Open the **Apama** tab.
6. Click **Create Apama Event Source**.
7. Set the properties for this event source. See table below.
8. Click **Save**.

The *Apama Event Source* is created and listed by alias name.

Table 3. 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 instance	yes	Alias with the pre-configured connection specification of a running **Apama** system (local or remote). See "Manage Apama Instances" on page 209 for details.
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

Property	Required	Description
		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. 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

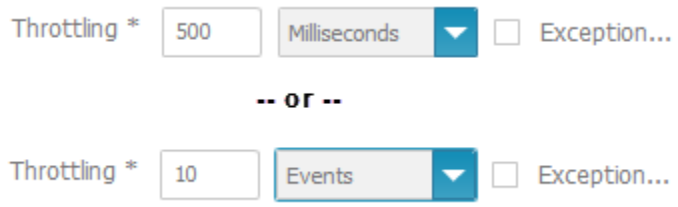
Property	Required	Description
		series (buckets) and stores the newer event in a new series (bucket). Default value: false
Capacity per dimension value	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set. Enter the maximum number 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 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)

Property	Required	Description
		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.
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.

Simple and Hybrid Throttling Strategies

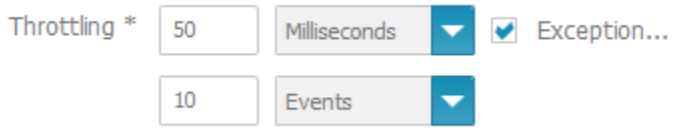
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.

5. Open the **Apama** tab.
6. Click the **Edit** icon to configure an Apama Event Source.
7. Set the properties for this event source. See table "Apama Event Source properties" on page 190 below.
8. Click **Save**.

The *Apama Event Source* is created and listed by alias name.

Table 4. 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 instance	yes	Alias with the pre-configured connection specification of a running **Apama** system (local or remote). See "Manage Apama Instances" on page 209 for details.
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.

Property	Required	Description
		■ 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. 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.

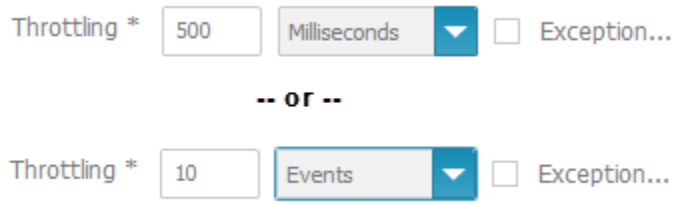
Property	Required	Description
		Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached. This option is clear by default which discards new events 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.
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

Property	Required	Description
		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
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.
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.

Simple and Hybrid Throttling Strategies

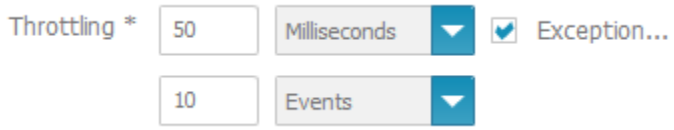
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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 properetie s.

Duplicate Apama Event Sources

You can duplicate an existing Apama Event Source.

To duplicate an Apama Event Source:

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.
5. Open the **Apama** tab.
6. 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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page is displayed.
5. Open the **Apama** tab.
6. Click the **Delete** icon to delete a specific Apama Event Source.
7. Click **Save**.

The selected Apama Event Source is deleted from the list.

Share Apama Event Sources

You can share Apama Event Sources with particular users and user groups so that these have access to *Apama Event Services*.

You have administration privileges.

Regardless of the share, users with administration privilege can access all Apama Event sources.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**.
4. Open the **Apama** tab.
5. Click the **Edit event service permissions** icon of the Apama Event Service you want to share.
6. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
7. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
8. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the *Apama Event Services* is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

9. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

10. Click **Ok**.

Your changes are applied.

Manage DES Event Sources

You can manage your DES Event Sources in the **Admin console**.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **Digital Event Services** tab.
6. Select further steps:
   - "Identify the DES repository directory" on page 197

Identify the DES repository directory

The DES Type Repository 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 DES Type Repository directory.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **Digital Event Services** tab.
6. Click **DES Type Repository**.
7. Enter the path to the local DES type repository in the **DES Type Repository Home Directory** field.
8. Click **Save**.

Your changes are applied.

Create DES Event Source

"You have identified the DES repository directory" on page 197.

Once you have identified the Event Type Store directory you can register subscriptions with the Event Bus. This creates *DES Event Services* that hold published events in memory and a corresponding event mashable in MashZone NextGen.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.

3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **Digital Event Services** tab.
6. Click **Create DES Event Source**.
7. Set the properties for this event source. See table below.
8. Click **Save**.

The DES Event Service is created and listed by alias name.

Table 5. DES Event Services 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.
Event type	yes	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.
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`.

Property	Required	Description
		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
Capacity per dimension value	conditional	Available only when **Strategy** is set to `BUFFER` and required when the **Consider dimension** option is set.

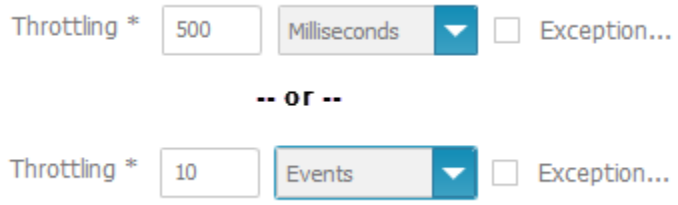
Property	Required	Description
		Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value. Default value: 10 The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000.
Event ID 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 events are received and change the number, if needed.

Property	Required	Description
		See Example below.
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.

Simple and Hybrid Throttling Strategies

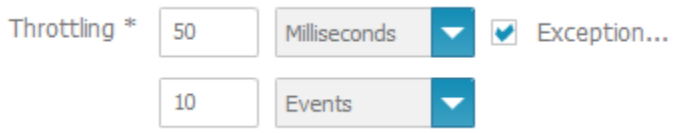
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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 DES Event Sources

You can edit already existing DES Event Sources.

Note: Changes in DES connection properties can immediately affect data feed calculations so that they may not execute properly.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **Digital Event Services** tab.
6. Click the **Edit** icon to configure a specific DES connection.
7. Set the properties for this event source:

Table 6. DES Event Services 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.
Event type	yes	Select the type of the event this event source should subscribe to.

Property	Required	Description
		The XML schema files for these event types must exist in the Event Type Store directory.
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. 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.

Property	Required	Description
		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.
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

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.
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.

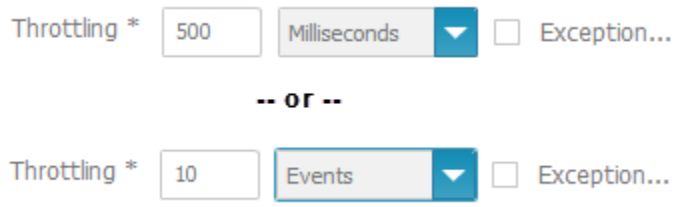
8. Click **Save**.

Your changes are applied.

Simple and Hybrid Throttling Strategies

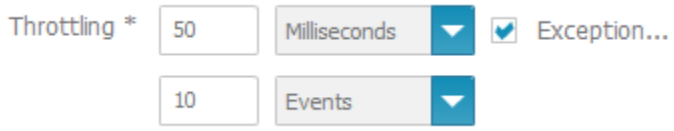
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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 DES Event Sources

You can duplicate existing DES Event Sources.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.

5. Open the **Digital Event Services** tab.
6. Click the **Copy** icon to duplicate a specific DES Event Source.

The selected DES Event Source is duplicated and listed with the prefix **copy_** in the **Alias**.

Delete DES Event Sources

You can delete existing DES Event Sources.

Note: Deleting DES Event Sources may cause data feeds to fail.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the **Digital Event Services** tab.
6. Click the **Delete** icon to delete a specific DES Event Source.

The selected DES Event Source is deleted from the list.

Share DES Event Sources

You can share DES Event Sources with particular users and user groups so that these have access to *DES Event Services*.

You have administration privileges.

Regardless of the share, users with administration privilege can access all DES Event sources.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**.
4. Open the **Digital Event Services** tab.
5. Click the **Edit event service permissions** icon of the DES Event Service you want to share.
6. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.

7. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
8. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the *DES Event Services* is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

9. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

10. Click **Ok**.

Your changes are applied.

Activate DES in MashZone NextGen

To use Digital Event Services (DES), a valid DES license file must be present in the MashZone NextGen installation.

The default path to the license is <MashZone NextGen installation>/common/DigitalEventServices/license/license.xml. After installation, a 30-days trial license is present in this location. To use DES, you must replace it by a valid license after 30 days.

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.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Open the event source tab **EDA**, **Digital Event Services**, 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. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**. The **Event Service** page will be displayed.
5. Click **Restart all** to restart all event source instances of the selected event source type.

All event sources are restarted.

Manage Apama Instances

By creating an Apama instance (Apama correlator) you can specify the connection to an Apama system.

You can create, edit and delete instances.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Apama instances**.
5. Select further steps:
   - "Create Apama Instances" on page 210
   - "Edit Apama Instances" on page 210
   - "Delete Apama Instances" on page 211

Create Apama Instances

By creating an Apama instance (Apama correlator) you can specify the connection to an runningApama system.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Apama instances**.
5. Click **Create**.
6. Set the properties of the Apama instance. See table below.
7. Click **Save**.

The Apama instance is created and listed by alias name.

Table 7. Apama instance properties

Property	Required	Description
Alias	yes	Enter a unique name for this Apama instance.
Host	yes	Host name to the running Apama system (local or remote)
Port	yes	Port number of the running Apama system (local or remote)

Edit Apama Instances

You can edit an already existing Apama instances.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Apama instances**.

5. Click the **Edit** icon to configure an Apama instance.
6. Set the properties of the Apama instance. See table below.
7. Click **Save**.

Your changes are applied.

Table 8. Apama instance properties

Property	Required	Description
Alias	yes	Enter a unique name for this Apama instance.
Host	yes	Host name to the running Apama system (local or remote)
Port	yes	Port number of the running Apama system (local or remote)

Delete Apama Instances

You can delete Apama instances.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Apama instances**.
5. Click the **Delete** icon to delete a specific Apama instance.

The selected Apama instance is deleted from the list.

Manage Apama Event Targets

An Apama event target specifies an Apama system that can receive events sent by MashZone NextGen.

You can create, edit and delete Apama event targets.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.

2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.
5. Open the **Apama** tab.
6. Select further steps:
   - "Create Apama Event Targets" on page 212
   - "Edit Apama Event Targets" on page 213
   - "Delete Apama Event Targets" on page 214
   - "Share Apama Event Target" on page 214

Create Apama Event Targets

By creating an **Apama** event target you can specify an **Apama** system as target, receiving events from MashZone NextGen.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.
5. Open the **Apama** tab.
6. Click **Create Apama Event Target**.
7. Set the properties for this event target. See table below.
8. Click **Save**.

The Apama event target is created and listed by alias name.

Table 9. Apama event target properties

Property	Required	Description
Alias	yes	Enter a unique name for this event target.
Apama instance	yes	Alias with the pre-configured connection specification of a running **Apama** system (local or remote). See "Manage Apama Instances" on page 209 for details.

Property	Required	Description
Event type	yes	Click **Refresh** to update the list of **Apama** event types. Only event types are available that are present in the Apama instance selected. Select the type of the event this event target should subscribe to. Event types including not supported data types by MashZone NextGen are also available; these event types can be used for sending events, but the offending fields are not usable for data assignment to dashboard components.

Edit Apama Event Targets

You can edit an already existing Apama event target.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.
5. Open the **Apama** tab.
6. Click the **Edit** icon to configure an Apama event target.
7. Set the properties for this event target. See table below.
8. Click **Save**.

Your changes are applied.

Table 10. Apama event target properties

Property	Required	Description
Alias	yes	Enter a unique name for this event target.
Apama instance	yes	Alias with the pre-configured connection specification of a running **Apama** system (local or remote). See "Manage Apama Instances" on page 209 for details.

Property	Required	Description
Event type	yes	Click **Refresh** to update the list of **Apama** event types. Select the type of the event this event target should subscribe to. Event types including not supported data types by MashZone NextGen are also available; these event types can be used for sending events, but the offending fields are not usable for data assignment to dashboard components.

Delete Apama Event Targets

You can delete Apama event targets.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Event Service** to expand this section of the Administration menu.
4. Click **Event Service**.
5. Open the **Apama** tab.
6. Click the **Delete** icon to delete a specific Apama event target.

The selected Apama event target is deleted from the list.

Share Apama Event Target

You can share Apama event targets with particular users and user groups so that these have access to Apama event targets.

You have administration privileges.

Regardless of the share, users with administration privilege can access all Apama event targets.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**.
4. Open the **Apama** tab.

5. Click the **Edit permissions** icon of the Apama event target you want to share.
6. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
7. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
8. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the *Apama Event Services* is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

9. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **View** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Edit** privilege has access to the relevant alias in the data source operator.

10. Click **Ok**.

Your changes are applied.

6 Process Performance Manager 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 as a source of data for MashZone dashboards and data feeds.

Note: MashZone NextGen is compatible with PPM version 10.0 or above.

Manage PPM Connections

You can manage your PPM Connections in the **Admin console**.

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **PPM connections** to expand this section of the Administration menu.
4. Click **PPM connections**.
5. Follow the procedure of the remaining steps:
   - "Create PPM Connections" on page 218
   - "Edit PPM Connections" on page 220
   - "Delete PPM Connections" on page 221
   - "Share PPM connections" on page 221

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 10.0 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

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **Connections** to expand this section of the Administration menu.
4. Click **PPM connections**.
5. Click **Create**.
6. 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.
7. To retrieve the connection data from the URL of a favorite from PPM:
   a. Click **Retrieve data**.
   b. Enter the URL of the PPM favorite that you copied earlier in the **URL** field.
   c. 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.
8. 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.

      For safety reason, we recommend to use the HTTPS protocol.
   b. In the **Host** field, enter the fully qualified domain name of the PPM load balancer.
   c. In the **Port** field, enter the port number of the PPM load balancer.
   d. Specify the PPM client name of your PPM connection in the **Client** field.
9. Click **Check availability** to verify that the data is correct and that the PPM client is available.
10. 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

Procedure

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **PPM connections** to expand this section of the Administration menu.
4. Click **PPM connections**. A list of all available PPM connections will be displayed.
5. Click the **Edit** icon to configure a PPM connection.
6. The **Alias** field of an already configured **PPM connection** is not editable. The connection data is saved under this alias.
7. To retrieve the connection data from the URL of a favorite from PPM:
   a. Click **Retrieve data**.
   b. Enter the URL of the PPM favorite that you copied earlier in the **URL** field.
   c. 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.
8. 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.

      For safety reason, we recommend to use the HTTPS protocol.

b. In the **Host** field, enter the fully qualified domain name of the PPM load balancer.

c. In the **Port** field, enter the port number of the PPM load balancer.

d. Specify the PPM client name of your PPM connection in the **Client** field.

9. Click **Check availability** to verify that the data is correct and that the PPM client is available.
10. Click **Save**.

Your changes are applied.

Delete PPM Connections

You can delete existing PPM connections.

Note: Deleting PPM connections may cause data feeds to fail.

To delete PPM connections:

1. In the program bar click the user name by which you are logged in to MashZone NextGen.
2. Click **Admin Console**.
3. Click **PPM connections** to expand this section of the Administration menu.
4. Click **PPM connections**. A list of all available PPM connections will be displayed.
5. Click the **Delete** icon to delete a PPM connection.
6. Confirm the deletion.

The selected PPM connections are deleted from the list.

Share PPM connections

You can share PPM connections with particular users and user groups so that these have access to PPM server.

You have administration privileges.

Regardless of the share, users with administration privilege can access all PPM connections.

Regardless of the share, users with administration privilege can access all EDA Event sources.

Procedure

1. Click **Admin Console** in the user menu of the program bar.

2. Click **PPM Connections** to expand this section of the Administration menu.
3. Click **PPM Connections**.
4. Click the **Edit PPM alias permissions** icon of the PPM connection you want to share.
5. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
6. Click **Show MashZone NextGen default groups** to show only default MashZone NextGen users or user group in the **Search results** field.
7. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the PPM connection is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

8. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

9. Click **Ok**.

Your changes are applied.

7 MashZone NextGen Explorer Integration

Create link to MashZone NextGen Explorer

You can create a link to open MashZone NextGen Explorer from the MashZone NextGen landing page.

You need to create a specific URL alias to configure the link.

Procedure

1. Create a URL alias in the **Admin console**, see"Create URL alias" on page 245.
2. Enter **MzNG Explorer** as alias name.
3. Enter the URL referring to your MashZone NextGen Explorer installation in the following form.

   `<computer name>.<dnsdomain>:<port>`

   The default port number is **9002**.

   e.g., localhost:9002

Your settings are applied. The **Explorer** link is automatically provided in program bar of MashZone NextGen landing page.

Displaying MashZone NextGen Explorer analyses in MashZone NextGen

You can publish your MashZone NextGen Explorer analyses and displaying them in MashZone NextGen.

Pre-requisites: You have installed MashZone NextGen and MashZone NextGen Explorer, and MashZone NextGen Server is running.

When publishing your MashZone NextGen Explorer analysis in MashZone NextGen the analysis data is transferred to MashZone NextGen and the analysis is automatically displayed in a corresponding dashboard. The analysis layout is adapted to the MashZone NextGen dashboard requirements. For each filter specified in MashZone NextGen Explorer a corresponding combobox is provided in MashZone NextGen that you can use to filter the analysis data.

In addition, a data feed including the **MZNG Explorer** data source operator is created with the data source settings of the MashZone NextGen Explorer analysis. See **MZNG Explorer** data source operator for details.

You can edit and use the dashboard and the data feed of your analysis in a similar way to other MashZone NextGen dashboard components.

Details on how to publish MashZone NextGen Explorer analyses in MashZone NextGen can be found in the MashZone NextGen Explorer User Guide.

8 webMethods Business Console Integration

MashZone NextGen can be easily embedded in webMethods Business Console using a native Business Console gadget.

The gadget is called MashZone NextGen and can be found in the Business Console **Common** section.

Detailed information on how to use webMethods Business Console can be found in the documentation **Working with webMethods Business Console**.

In order to access a MashZone NextGen dashbaord, the **Dashboard URL** must be provided in the gadget settings. The URL must contain the MashZone NextGen dashboard GUID as an URL parameter.

Example

http://sbrvpresto4.eur.ad.sag:8080/mashzone/hub/dashboard/dashboard.jsp?guid=e35c1619-0b06-42ac-b343-b16e7d5dcc12

In the section **UI Settings** the gadget height, title and border can be specified.

The section **Data Mapping** specifies the parameters required for the communication between MashZone NextGen dashboards and Business Console gadgets.

- **Mapping Id**: Identifier used in Business Console for gadget to gadget communication. The **Mapping Id** is needed to identify and map the data send from one gadget to the data structure of another gadget. In case of two MashZone NextGen gadgets, exchanging data, the mapping can be used to take a selection value from one embedded MashZone NextGen widget and use it as selection value in the other MashZone NextGen widget.
- **Widget Id**: Specifies the external identifier of the MashZone NextGen widget to communicate with.
- **Widget Parameter**: Specifies a measure or dimension name used in the MashZone NextGen widget.
- **Default Value**: Optionally, it is possible to define a default value, that is used for example as a default selection for the MashZone NextGen widget after loading the gadget in Business Console.

All data required can be found in MashZone NextGen, see **Use dynamic URL selection** for details.

Authentication

You can integrate MashZone NextGen under My webMethods in an SSO scenario by SAML (Security Assertion Markup Language).

MashZone NextGen can accept SAML tokens for authentication in a SSO environment.

See "Authentication with Single Sign-On Solutions" on page 61 for details.

A BASE64 encoded SAMLToken is expected. Since it is send via URL it needs to be URL encoded before.

Example URL

http://sbrvpresto4.eur.ad.sag:8080/mashzone/hub/dashboard/dashboard.jsp?appheader=false&guid=64545b4f-d150-4241-a858-2304eea23684 &SAMLToken=<URL encodedBASE64 encoded token>

Configuration

To enable access to MashZone NextGen you need to list the URL(s) of the webMethods Business Console server(s) in the Content Security Policy of MashZone NextGen.

The content security settings are done in the server configuration file applicationContext-security-filters.xml by adding filters for X-Frame-Options and Content Security Policies. The file is located in *<MashZone NextGen installation>* \apache-tomcat\webapps\mashzone\WEB-INF\classes.

applicationContext-security-filters.xml (abstract)

```
<beans:beans
    xmlns="http://www.springframework.org/schema/security"...>
    ...
        <http pattern="/hub/login.html" security="none"/>
    <http pattern="/**/*.jsp" use-expressions="false"
          authentication-manager-ref="authenticationManager"
          entry-point-ref="mzngAuthenticationEntryPoint">
          <anonymous enabled="false"/>
          <headers>
            <!--frame-options policy="SAMEORIGIN"/-->
            <frame-options policy="ALLOW-FROM" strategy="whitelist"
             value="http://BCServerHostA:BCServerPortA,
                    http://BCServerHostB:BCServerPortB,..."/>
            <!--content-security-policy policy-directives="frame-ancestors 'self'"/-->
            <content-security-policy policy-directives="frame-ancestors 'self'
             http://BCServerHostA:BCServerPortA, http://BCServerHostB:BCServerPortB,.../>
          </headers>
        <csrf token-repository-ref="csrfTokenRepository"
         request-matcher-ref="skipHttpAuthCsrfMatcher"/>
    </http>
    <http pattern="/**/*.html" use-expressions="false"
          authentication-manager-ref="authenticationManager"
          entry-point-ref="mzngAuthenticationEntryPoint">
          <intercept-url pattern="/**/*.html"
                         access="IS_AUTHENTICATED_ANONYMOUSLY"/>
          <anonymous enabled="false"/>
          <headers>
            <!--frame-options policy="SAMEORIGIN"/-->
            <frame-options policy="ALLOW-FROM" strategy="whitelist"
             value="http://BCServerHostA:BCServerPortA,
                    http://BCServerHostB:BCServerPortB,..."/>
            <!--content-security-policy policy-directives="frame-ancestors 'self'"/-->
            <content-security-policy policy-directives="frame-ancestors 'self'
             http://BCServerHostA:BCServerPortA, http://BCServerHostB:BCServerPortB,..."/>
          </headers>
    </http>
```

```
    ...
</beans:beans>
```

Outbound API

MashZone NextGen provides an outbound API to pass data from MashZone NextGen dashboards to an embedding system, e.g., an external web application like webMethods Business Console.

See **Post data** for details.

Inbound API

By using iFrame MashZone NextGen can be used as a component in external products, e.g., webMethods Business Console. As embedded component MashZone NextGen is enabled to send data via outbound API (Post data) to the embedding system and receive data via inbound API (URL selection) from the embedding system.

See "Embedding MashZone NextGen in external system environments" on page 101 for details.

9 MashZone NextGen 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: The MashZone NextGen repository is initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move the repositories to a robust and compatible solution.

Configuration and administration tasks for these two repositories include:

- Move the MashZone NextGen repository 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 Repository Connection Pool" on page 231
- 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 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 Repository Connection Pool

In addition to basic connection configuration, you can configure the connection pools for the MashZone NextGen 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 Tomcat Datasource Properties.

To tune the connection pool, you update properties in the `<Resource>` element for the MashZone NextGen repository in the *MashZoneNG-install* /apache-tomcat/conf/context.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 repository to a robust database solution" on page 19 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* /mashzone/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 17 for instructions.

10 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

View MashZone NextGen Logs

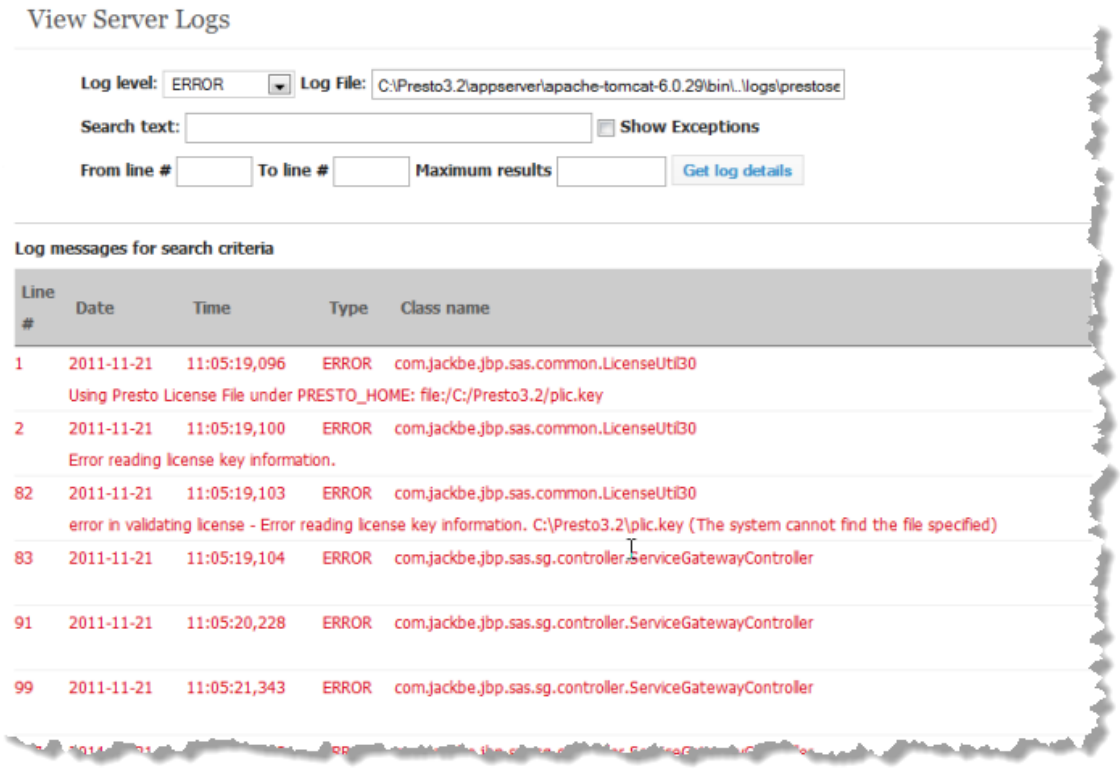
You can view two MashZone NextGen logs in MashZone NextGen Hub. See "View the MashZone NextGen Server Log" on page 236 and "View the Audit Log for a Mashable, Mashup or App" on page 237 for instructions. See also "MashZone NextGen Logging" on page 113 for links to additional logging options for MashZone 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.

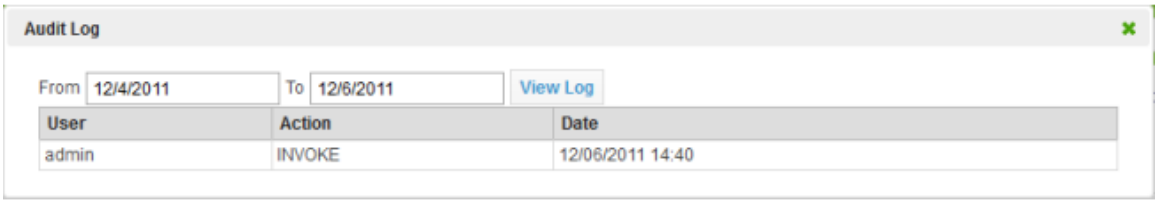
View the Audit Log for a Mashable, Mashup or App

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 115).

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 238, "Manage Pluggable Views" on page 239 and "Manually Changing the Default Version for Libraries" on page 240 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.

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 240 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.

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 240 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.

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 242 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 238 .

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* `/mashzone/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* `/mashzone/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 242 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 242 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 241 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 /mashzone/files` to the file name.

Manage resource directories

Resource directories hold file-based data sources, such as Excel spreadsheets, CSV or XML files.

The resource aliases can be used by the data source operators to read local files.

Create resource directory

To work with data sources in MashZone NextGen Feed Editor 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 NextGen 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.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **File resource** -> **File resource** to open the resource alias page.
3. Click **Create**.
4. Give the directory an alias name of your choice in the **Resource directory** input box.

   You cannot modify the alias name later.
5. Enter the Path of the new resource directory.
6. Click **Add resource**.

The new resource directory is created and is displayed in the list with the specified alias.

Change resource directory

You can adapt the path of already existing resource directories.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **File resource** -> **File resource** to open the resource alias page.
3. Click the **Edit resource alias** icon of the resource you want to edit.
4. Enter the **Path** of the resource directory.
5. Click **Save resources**.

Your changes are applied.

Delete resource directory

You can delete existing resource directories.

1. Click **Admin Console** in the user menu of the program bar.
2. Click **File resource** -> **File resource** to open the resource alias page.
3. Click the **Delete resource alias** icon of the resource you want to delete.
4. Click **Yes**.

The directory selected is deleted from the list.

Share resource directory

You can share resource directories with particular users and user groups so that these have access to the directory content.

You have administration privileges.

Regardless of the share, users with administration privilege can access all resource directories.

Procedure

1. Click Admin Console in the MashZone NextGen Hub main menu.
2. Click **Resource alias** -> **Resource alias** in the Administration menu.
3. Click the **Edit resource permissions** icon of the resource you want to share.
4. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.

5. Click **Show MashZone NextGen default 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.

   Note: By default, the owner of the resource directory is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

7. Drag an user or user group from the **Search result** field and drop it into the **Principals with permissions** field.

   Note: By default, the owner of the *Apama Event Services* is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

8. Click **Ok**.

Your changes are applied.

Manage URL aliases

You can manage your URL aliases in the **Admin console**.

Using an URL alias is always recommended to shorten the link used in ,e.g., dashboards and data feeds. You have to enter the path where the data are stored only, and not the complete URL. The resource aliases can be used by the data source operators to read local files.

Create URL alias

You can create URL aliases to shorten a link used in ,e.g., dashboards and data feeds.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **URL aliases** -> **URL aliases** to open the URL alias page.
3. Click **Create**.
4. Give the URL an alias name of your choice in the **Alias** input box.

   You cannot modify the alias name later.

5. Enter the **URL**.
6. Activate the **Use basic authentication** option if an authentication is require for using the URL.

   a. Enter an **Username**.

b. Enter the **Password** associated with the username.

7. Click **Add alias**.

The new URL alias is created and is displayed in the URL alias list.

Change URL alias

You can adapt the URL and the authentication credentials of already existing URL aliases.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **URL aliases** -> **URL aliases** to open the URL alias page.
3. Click the **Edit URL alias** icon of the URL alias you want to edit.
4. Make your changes.
5. Click **Save alias**.

Your changes are applied.

Delete URL alias

You can delete existing URL aliases.

1. Click **Admin Console** in the user menu of the program bar.
2. Click **URL aliases** -> **URL aliases** to open the URL alias page.
3. Click the **Delete URL alias** icon of the URL alias you want to delete.
4. Click **Yes**.

The URL alias selected is deleted from the list.

Share URL alias

You can share URL aliases with particular users and user groups so that these have access to the directory content.

You have administration privileges.

Regardless of the share, users with administration privilege can access all URL aliases.

Procedure

1. Click **Admin Console** in the user menu of the program bar.
2. Click **URL aliases** -> **URL aliases** to open the URL alias page.
3. Click the **Edit URL alias permissions** icon of the alias you want to share.

4. Enter a therm in the search field and click **Search**. Clicking on **Search** without any input values fetches all users and groups.
5. Click **Show MashZone NextGen default 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.

   Note: By default, the owner of the URL alias is already present in the **Principals with permissions** list . This owner is non editable and cannot be removed from the list.

7. Activate or deactivate the **Display** or **Usage** privileges of a user or user group.

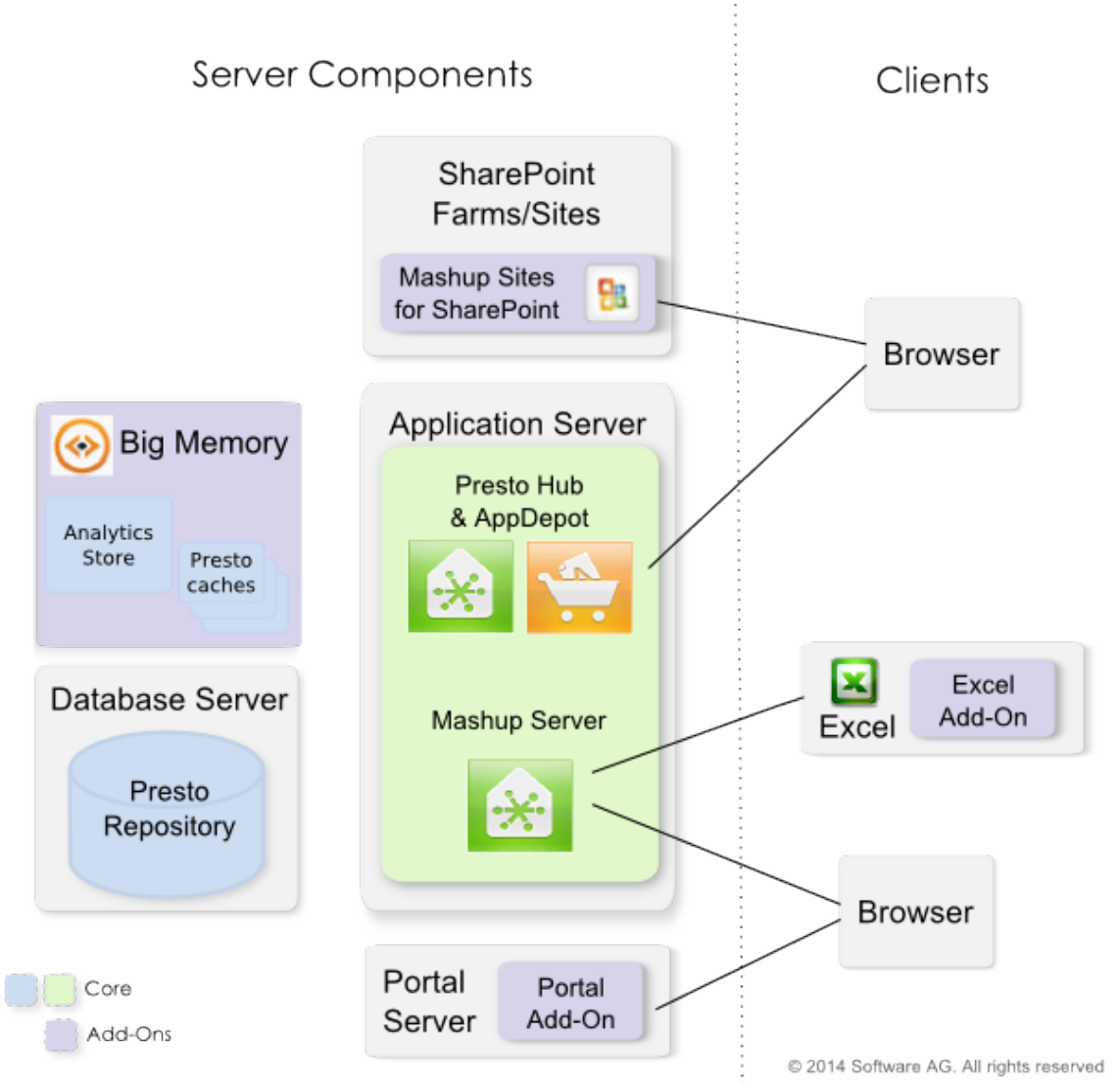
   A user or user group with **Display** privilege can see the relevant source data in the data feed or dashboard. A user or user group with the **Usage** privilege has access to the relevant alias in the data source operator.

8. Click **Ok**.

Your changes are applied.

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 MashZone NextGen Artifacts and Other Metadata" on page 249.



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 repository to a robust database solution" on page 19 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.

To deploy multiple unclustered servers, see "Deploying Multiple MashZone NextGen Servers in One Host" on page 274. To deploy MashZone NextGen servers in clusters, see "Clustering MashZone NextGen Servers" on page 275 for requirements and links.

Deploying MashZone NextGen Artifacts and Other Metadata

You deploy specific artifacts and 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 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 Macros" on page 256
   - "Exporting Pluggable Views or Libraries" on page 259
   - "Exporting MashZone NextGen Global Attributes" on page 262
   - "Exporting Users, User Metadata and Groups" on page 262

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 280 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.

   See "Manage data sources and drivers" on page 145 for instructions.

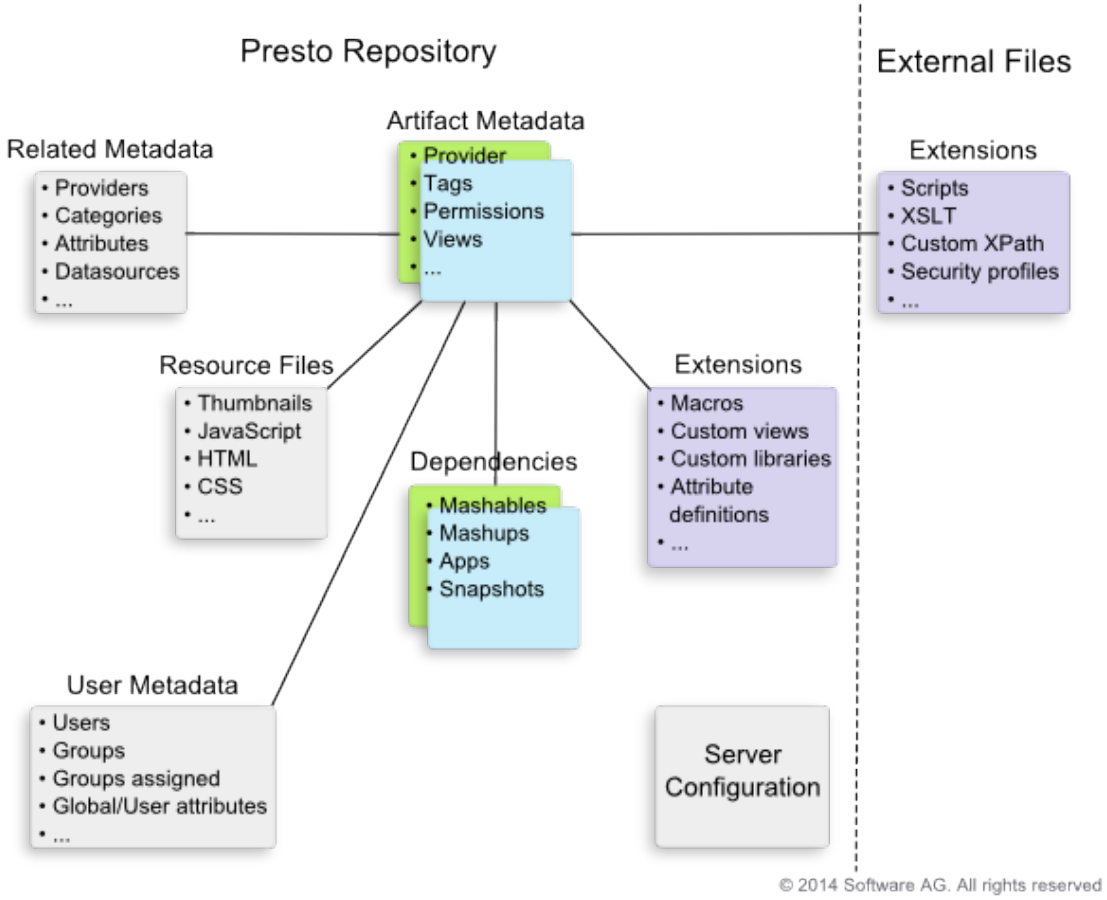
Deploying MashZone NextGen Artifacts and Other Metadata

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 255
   - "Exporting Macros" on page 256
   - "Exporting App MetaData" on page 258
   - "Exporting Pluggable Views or Libraries" on page 259
   - "Exporting MashZone NextGen Global Attributes" on page 262
   - "Exporting Users, User Metadata and Groups" on page 262

   The data that is exported and known limitations for these commands include:

Table 11. 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 280 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 145 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 266
   - "Importing Mashable or Mashup MetaData" on page 265
   - "Importing Pluggable Views or Libraries" on page 269
   - "Importing App Metadata" on page 267
   - "Importing MashZone NextGen Global Attributes" on page 271
   - "Importing Users, User Metadata and Groups" on page 271

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 11" on page 251 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 17 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/mashzone/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 17 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/mashzone/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 249 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 17 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/mashzone/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 8. Examples" on page 261 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.

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.

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 17 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/mashzone/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/mashzone/edge/api -u Administrator -w manage
```

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 17 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/mashzone/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).

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 17 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/mashzone/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 dashboards

You can export your MashZone NextGen dashboards.

Exporting dashboards creates a zip file that you can use to create a backup or to import your dashboards into another MashZone NextGen installation.

Procedure

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter this command:

```
padmin exportDashboard -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 only the first dashboard found will be exported.

-i "id=43243244434432": The dashboard ID (GUID) is unique in the MashZone NextGen system.

-i "all": Exports all dashboards for that user.

- *-f output-file* : Optional path and name for the export. If omitted, an output zip file is created in the folder in which 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
- *-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/mashzone/esd/api`.
- *-u username* : MashZone NextGen user name 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 activate verbose logging.
- *-o*: Optional flag to overwrite an existing export file.

Once the export command completes successfully, you can use the output file to import dashboards into MashZone NextGen.

Permissions for each dashboard were automatically stored in the zip file. If no permissions are assigned to the dashboard, the permission file saved is empty.

The zip file also includes information about the dashboard creator.

Exporting data feeds

You can export your MashZone NextGen data feeds.

Export creates an export file that you can use to import data feeds to MashZone NextGen.

Procedure

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter this command:

```
padmin exportFeed -i identifier [-f output-file] [-l prestoURL]
-u username -w password [-v] [-o]
```

- -i *identifier* : mandatory data feed identifier. It can be "id=", "name=" or "all", enclosed in quots.

  -i "name=feedname": If there are multiple data feeds with the same name then only the first founded data feed will be exported.

-i "id=43243244434432": The data feed id (Guid) is unique in the MashZone NextGen system.

-i "all": Export of all data feeds for that user.

- *-f output-file* : an optional path and name for the export file to put data feeds. 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 datafeed-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/mashzone/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 data feeds to MashZone NextGen.

Permissions for each data feed were automatically stored into the zip file. If there are not any permissions assigned to the data feed an empty permission file is stored.

There is also an information about the data feed creator stored in the zip file.

Example

```
padmin exportFeed -l http://localhost:8080/mashzone/esd/api -f feedDefinition.zip
 -u Administrator -w manage -i "id=MyFeed"
```

This created zip file "feedDefinition.zip" contains all information of the data feed "MyFeed" incl. definition and permissions.

Importing Mashable or Mashup MetaData

you must have a mashable or mashup export file to import. See "Exporting Mashable and Mashup MetaData" on page 255 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 17 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/mashzone/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 256 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 17 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/mashzone/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 258 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 265 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 17 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/mashzone/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 259 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.

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 12. Example" on page 270 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.

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 17 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

   - *-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/mashzone/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
```

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 262 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 17 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/mashzone/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 262 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 17 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/mashzone/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.

Importing dashboards

You can import dashboards into MashZone NextGen.

The dashboards are saved in a zip containing the dashboard definition, resource policy, and dashboard permissions, etc.. If you import a dashboard including permissions the creator of the dashboard can view and edit the dashboard. The importer of a dashboard

automatically becomes the creator of the dashboard if the dashboard is imported without permissions.

Procedure

1. Open a command window and move to the <*MashZoneNG-installation*> /prestocli/bin folder.
2. Enter this command:

```
padmin importDashboard [-l prestoURL] -f input-file
-p importPermissions -u username -w password
[-v] [-o]
```

   - *-f input-file* : Path and name of the import zip file.
   - -p *importPermissions* : Imports the resource policy and permissions saved in the import zip file.

     The importer of a dashboard automatically becomes the creator of the dashboard if the dashboard is imported without permissions. And only administrators can see and work with the dashboards imported.
   - *-o*: is optional. Allows overwriting an existing dashboard in MashZone NextGenDashboard.
   - *-l prestoUrl* : 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 results in `http://localhost:8080/mashzone/esd/api`.
   - *-u username* : is the MashZone NextGen user name 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 the MashZone NextGen Dashboard component.

If you have imported dashboards from Presto 3.9 into MashZone NextGen, save the imported dashboards in edit mode of the Dashboard component before you display them in view mode. Otherwise, an error message is displayed.

Importing data feeds

You can import data feeds to MashZone NextGen.

The data feeds are saved in a zip file that contains among other things the data feed definition, resource policy and data feed permissions. If you import a data feed including the permissions then the creator of the data feed can view and edit the data feed. Importing data feeds without the relevant permissions makes the importer automatically to the creator of these data feeds.

Procedure

1. Open a command window and move to the *MashZoneNG-install* /prestocli/bin folder.
2. Enter this command:

```
padmin importFeed [-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 data feeds without permissions makes the importer automatically to the creator of these data feeds and the data feeds has no explicit permissions which means that only administrators can see and work with the data feeds .
   - *-o*: is optional. Allows to overwrite an existing data feeds.
   - *-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/mashzone/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 data feeds in MashZone NextGen Feed Editor.

Example

```
pAdmin importFeed -l http://localhost:8080/mashzone/esd/api -f feedDefinition.zip
-u Administrator -w manage -o
```

With this command the content of the data feed file "feedDefinition.zip " will be imported to MashZone NextGen.

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.

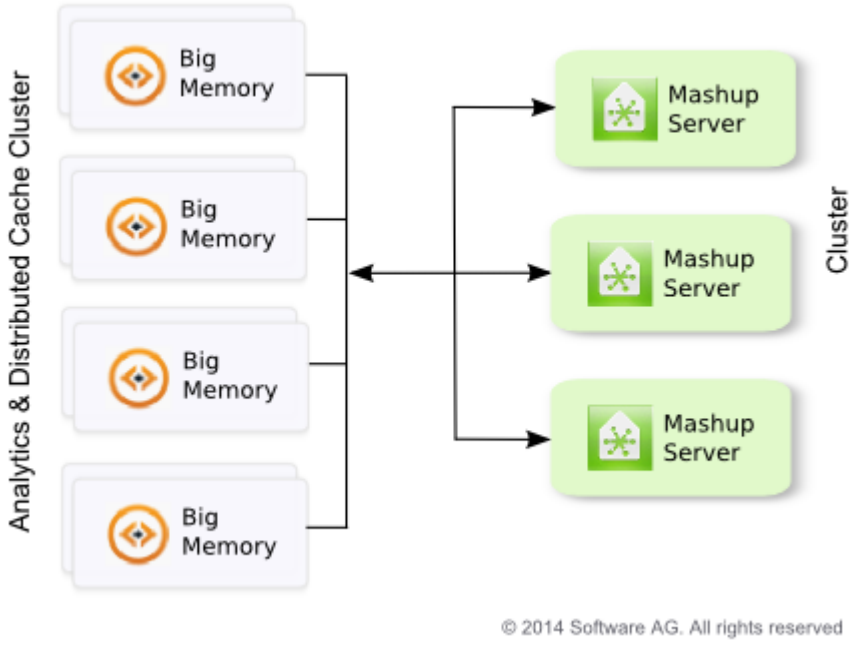
- To host multiple, independent servers, simply install each *being sure* to change the ports assigned to each MashZone NextGen Server, MashZone NextGen Repository and the administration port for Tomcat.

  Note: You can also create clusters of MashZone NextGen Servers to provide load balancing. See "Clustering MashZone NextGen Servers" on page 275 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.

A common architecture for MashZone NextGen clusters looks something like this:



See "Setting Up a New Cluster" on page 275 or "Adding New Members to an Existing Cluster" on page 277 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 280: 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 278: 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 127 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 277 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 249 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 279 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 281 for a list of files and folders to copy.
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 281 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 278 with cluster members, typically when you are creating new environments. Or you can "Share an Existing MashZone NextGen Repository" on page 279 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-install* /apache-tomcat/lib folder on all cluster nodes.

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 repository to a robust database solution" on page 19 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 280 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 127 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 *MashZoneNG-install* /apache-tomcat/lib folder for the new MashZone NextGen Server cluster member.

   b. Copy the rdsJdbc.properties file from the *web-apps* /mashzone/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 280 for more information on shared configuration for clusters.

2. 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 127 for more information and instructions.

3. If the MashZone NextGen Repository is hosted in Microsoft SQL Server, MySQL or Oracle, change the repository JAR in the MashZone NextGen Server.

4. 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-tomcat/mashzone/WEB-INF/classes for class, configuration and extension files
- *MashZoneNG-install* /apache-tomcat/mashzone/WEB-INF/lib and *MashZoneNG-install* /apache-tomcat/mashzone/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.
   - Built-in and user-defined functions for use in RAQL queries for MashZone NextGen Analytics.

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 281 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 *tomcat-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 *tomcat-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 281 and "MashZone NextGen Extensions" on page 284 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 283 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-tomcat/mashzone/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	

File	Description and Configuration	Default Location
	the path to the deployed web app home folder.	
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 properties are also used in some specific cases. See "Move the MashZone NextGen repository to a robust database solution" on page 19 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 127 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 34 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* /mashzone/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 127 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 61 or "Authentication with Digital Certificates/SSL" on page 69 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 69 for more information.

File	Description and Configuration
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 repository to a robust database solution" on page 19 for more information.
log4j.properties	This file is updated automatically when you change logging configuration in the Admin Console. See "Configure Logging for the MashZone NextGen Server" on page 113 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 34 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
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` ■ Groovy scripts	*MashZoneNG-install* /apache-tomcat/mashzone/WEB-INF/classes or

File	Default Location
XSLT stylesheets called in mashups in EMML using the <xslt> statement.	*MashZoneNG-install* /apache-tomcat/mashzone/WEB-INF/lib (for JARs)
Custom XPath function classes used in mashups in EMML.	
Local copies of WSDL files used for WSDL web services.	
Custom security profile classes used with mashables.	
Custom certificate validation classes for certificate authentication. See "Configure Additional Certificate Validation" on page 73 for details.	
Custom filter classes for single sign-on authentication. See "Implementing a Custom SSO Filter" on page 68 for details.	
Classes and third-party libraries for a user-defined function library to use with RAQL.	

MashZone NextGen Dashboard in a clustered scenario

You are able to use MashZone NextGen Dashboard in a clustered scenario.

The following chapters describe how to configure MashZone NextGen to use dashboards and data feeds in a multiple master-client scenario.

Preliminary

Before you can configure MashZone NextGen using in a clustered scenario you have to perform the following steps.

Procedure

1. Install at least two regular MashZone NextGen instances on two different machines.

   Software AG Installer enables you to install MashZone NextGen. Detailed information on how to use Software AG Installer is available in the documentation **Using the Software AG Installer**.

2. Connect all instances to the same central database according to section "Move the MashZone NextGen repository to a robust database solution" on page 19.
3. Remove rtbs from folder <*MashZoneNG_installation*> /apache-tomcat/webapps on all slave nodes.

The preliminary for configuring MashZone NextGen are completed.

Configuration

The following chapters describe the relevant configurations of MashZone NextGen Dashboard in a clustered scenario.

Note: Please node that RTBS has to run as single instance on the master server of the cluster.

Real-Time Buffer Server (RTBS)

Configure the following parameters on your MashZone NextGen slave and master nodes.

Slave node

As the clustered scenario only works with a single RTBS instance on the master server, the right endpoint for calling the RTBS API has to be configured.

Edit the `rtbs.base.url` parameter in the presto.config file and set host name and port accordingly.

<*MashZoneNG-installation*> /apache-tomcat/webapps/mashzone/WEB-INF/classes/presto.config

```
rtbs.base.url=http://masternode:8080/rtbs/services/bufferService/
```

Edit the `mashzone.rtbs.url` parameter in the mashzone.properties file and set host name and port accordingly.

<*MashZoneNG-installation*> /apache-tomcat/webapps/mashzone/WEB-INF/mashzone.properties

```
mashzone.rtbs.url=http://masternode:8080/rtbs
```

Replace the host name `masternode` by your real server's host name and also set the right port in both cases.

Note: Calls to the RTBS API are server-to-server calls which usually happen behind the load balancer. In some cases it might be necessary to route these calls through the load balancer as well. In this case, make sure that there is only one instance of the RTBS and configure your load balancer accordingly. Set the load balancer's host name and port in the parameters mentioned above.

Master node

In most cases you won't have to configure anything regarding RTBS on your master node, as the RTBS instance runs on the same machine. Depending on your load balancer set-up, it might be required that `localhost` is replaced by the master node's real host name resp. by the name of the virtual host. In this case just follow the instructions under section **Slave node** and set host name and port accordingly.

Load balancer

"sticky sessions" mode

The load balancer must run in "sticky sessions" mode. That means that all subsequent requests belonging to the same session will be send to the same server that handled the first request. The load balancer must be configured accordingly.

WebSocket communication

MashZone NextGen uses sockJS for the client - server communication. Requests that end with "/websocket" have to be routed to MashZone NextGen nodes, using ws/wss as protocol. If the load balancer can not handle the websocket communication (e.g., HTTP header connection: upgrade, upgrade: websocket), sockJS uses different HTTP fallback protocols.

Fail-over mode

In fail-over mode, only one server handles client requests and the second server just runs in background. If the first server fails, the second one takes over all traffic and handles the requests until the main server is available again. In this mode, the two servers have to be more or less identical installations with both, RTBS servers running and both connected to the same database. RTBS configuration is as it would be in a single server scenario.

Note: The RTBS traffic must also be mounted to the balancer worker in this case, not to the worker node that is the master instance.

Customizing dashboards

MashZone NextGen dashboards can be customized by adding custom style templates for the dashboard application and the dashboard content. Additionally custom components can be created via the pluggable widget framework. If these options shall be applied in a clustered scenario, you must synchronize the relevant folders and restart MashZone NextGen on all nodes of the cluster.

Custom styles

By default, custom style templates available are stored in the following folders.

- *<MashZoneNG-installation>* /apache-tomcat/webapps/mashzone/hub/dashboard/assets/custom-look-and-feel/application

- <*MashZoneNG-installation*> /apache-tomcat/webapps/mashzone/hub/dashboard/assets/custom-look-and-feel/dashboard

To apply the custom templates on all cluster nodes, make sure that these folders are synchronized on all machines. Since the less files need to be compiled before the styles can be used, MashZone NextGen has to be restarted on all cluster nodes.

Custom widgets

By default, custom widgets available are stored in the following folders.

<*MashZoneNG-installation*> /apache-tomcat/webapps/mashzone/hub/dashboard/widgets/customWidgets

To make the custom widgets available on all cluster nodes, make sure that the folders is synchronized on all machines. In this case, restarting MashZone NextGen on all cluster nodes is required as well.

Using JDBC drivers

JDBC driver binaries have to be available on every cluster node to allow class loading in the JVM. Since MashZone NextGen version 9.10 the binaries are stored in the DB and restored in <*MashZoneNG-installation*> \apache-tomcat\webapps\mashzone\WEB-INF\config\db\jdbc on all cluster nodes if not available. Automatic class loading on demand works fine, so that no further steps have to be taken to make JDBC resources available in a clustered scenario.

Local file resources

Local file resources are not recommended and not supported in a clustered scenario due to synchronization issues.

Note: In a Windows landscape it might be possible to use such file resources by mapping the same network drive to the same network share. There may also be other file sharing mechanisms working in other OS landscapes, but URL based access is preferable.