# Installation and Administration Guide

Version 9.9

October 2015

This document applies to Presto Version 9.9 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: PR-IA-39-20150415**

# Table of Contents

# Preface

The Presto Installation and Administration Guide includes instructions to install Presto along with information for Presto administrators to configure and manage the Presto Server.

# 1 Presto Administration

Presto may be installed by itself or as part of Intelligent Business Operations (IBO).

Presto administrators install, configure and manage the:

- Presto Server
- Integrated MashZone Server
- Event Service
- Presto Repository and MashZone Repository

See the installation and administration documentation for more information.

The following table lists starting points to help you find the tasks and configuration options for administrators:

| | |
|---|---|
| Post-installation tasks | See "Getting Started with the Presto Server" on page 13 for required tasks.<br>See "What's Next" on page 50 for links to optional configuration tasks for the Presto Server and IBO. |
| Overview | See "What is Installed with Presto" on page 19. |
| Authentication, authorization and other security configuration | See "Presto Security" on page 53 for concepts and configuration tasks. |
| Other Presto Server configuration tasks | See "Presto Server Configuration" on page 79 and "Process Performance Manager (PPM) Integration" on page 189 for links to a variety of configuration tasks. |
| Integrated MashZone Server and Event Service configuration | See "Integrated MashZone Server Configuration and Administration" on page 147 and "Event Service Configuration and Administration" on page 159 for more information. |
| Presto Repository configuration | "Presto and MashZone Repositories" on page 197 |

| | |
|---|---|
| Administration, upgrade and deployment tasks | "Presto Server Administration" on page 203 |

# 2 Getting Started with the Presto Server

You install Presto 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 Presto include:

1. Start the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Login to Presto Hub:
   a. Open Presto Hub in a browser at `http://localhost:8080/presto`.

      If you used a different port number when you installed Presto or the Presto 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 Presto User Repository rather than an LDAP Directory to manage users and groups for Presto, it is a good practice to change the password for this default administrator account.
   a. Open your profile from the Presto 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 Presto User Repository, this default account is disabled once LDAP configuration is complete.
4. Set up a robust database to use as the Presto Repository.

   Presto is installed with Derby as an embedded database hosting the Presto 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 Presto and MashZone Repositories to a Robust Database Solution" on page 24 for instructions.

   For Presto database compatibility information, see "Presto and MashZone Database Compatibility" on page 18.
5. If you want Presto to use your LDAP Directory as the repository for user and group information, you must update configuration. See "Integrate Your LDAP Directory with Presto" on page 40 for instructions.
6. Configure the Integrated MashZone Server and the Event Service. See "Integrated MashZone Server Configuration and Administration" on page 147 and "Event Service Configuration and Administration" on page 159 for instructions.

7. If you have also installed Terracotta BigMemory and received your BigMemory license, add this license to Presto and configure Presto to work with BigMemory. See "Manage Licenses for Presto, Universal Messaging and BigMemory" on page 23 and "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for instructions.

In addition to these required steps, there are many other optional configuration tasks you may wish to complete. See "What's Next" on page 50 for links to optional tasks.

# Additional Presto System and Software Requirements

- For basic requirements to install Presto, see the *System Requirements for Software AG Products* guide. For additional requirements or requirements for Presto Add-Ons, see:
  - "Additional Recommendations for Presto" on page 15
  - "Recommendations for Presto Add-On for SharePoint (P4S)" on page 16
  - "Recommendations for the Presto Add-On for Excel" on page 16
  - "Recommendations for the Presto Add-On for Portals" on page 17
  - "Requirements for Optional Views" on page 17
- For information on other compatibility questions, see:
  - "Presto Browser and Mobile Device Compatibility" on page 17
  - "Presto and MashZone Database Compatibility" on page 18

## Additional Recommendations for Presto

In addition to the basic recommendations in the *System Requirements for Software AG Products* guide, you should also consider the following recommendataions for Presto:

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

  **Important:** The Presto Repository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution. See "Presto and MashZone Database Compatibility" on page 18 for more information.

- Architecture and memory requirements or recommendations include:
  - 64-bit architecture
  - 1G minimum of memory if only small to medium datasets are involved
  - 4G minimum of memory if large datasets are involved

**Important:** Actual memory and disk space requirements vary widely based on load, throughput, performance and other requirements unique to your environment. Please contact your Software AG representative for more information.

See also "Presto Browser and Mobile Device Compatibility" on page 17.

## Recommendations for Presto Add-On for SharePoint (P4S)

- A compatible version of SharePoint:

| Presto Add-On for SharePoint / Presto | Microsoft Office SharePoint 2007 | Microsoft SharePoint Server 2010 |
|---|---|---|
| P4S 3.0 + Presto 3.0 | yes | |
| P4S 3.1 + Presto 3.1 | | yes |
| P4S 3.1.1 + Presto 3.1.1 or greater | yes | yes |

  The Single Sign-on Service (2007) or Secure Store Service (2010) in SharePoint are required to use the Presto Add-On for SharePoint Single Sign-On feature. This significantly improves the user experience by removing or reducing login challenges. See Microsoft documentation for more details on single sign-on capabilities in SharePoint.

- Microsoft .NET Framework 3.5 (available at http://www.microsoft.com/downloads/details.aspx?FamilyId=333325fd-ae52-4e35-b531-508d977d32a6).
- An appropriate version of the Windows operating system and hardware. See Microsoft documentation for details.

  **Note:** Per Microsoft, SharePoint Server 2010 requires a 64-bit architecture.

In most installations, authentication between SharePoint and Presto uses the NTLM version 2 protocol.

## Recommendations for the Presto Add-On for Excel

- 1.7 megabytes (MB) of available hard disk space.
- Recommended hardware requirements for Windows Office 2007 or 2010 and the .NET Framework 2.0 or above.
- Microsoft Office 2007 or Office 2010 (for a 32 bit architecture) and any compatible Windows operating system.

> **Note:** Because of a known issue, the Presto Add-On for Excel is not currently compatible with Microsoft Office 2010 for 64 bit systems.

- Windows Office 2007 Primary Interop Assemblies.
- Microsoft .NET Framework 2.0 or above.

## Recommendations for the Presto Add-On for Portals

The Presto Add-On for Portals is a portlet producer that is compatible with any portal that supports the JSR-168 Portlet Specification or JSR-286 Portlet 2.0 Specification. The Presto Add-On for Portals has no additional system requirements beyond those for your portal.

## Requirements for Optional Views

Google Map views do not require an API key from Google, unless your usage exceeds Google limitations.

## Presto Browser and Mobile Device Compatibility

This release of Presto has been tested against the following browsers and devices. Presto tools and apps are compatible with the known exceptions noted here. See also "Requirements for Optional Views" on page 17 for additional requirements.

| Desktop | Mobile Device and OS |
|---|---|
| Internet Explorer 9.0 and 10.<br><br>There is one known exception. PrestoMashboard, the visual composer for workspace apps, automatically runs in IE8 forced compatibility mode when running in IE 9.0. | **Note:** Currently Presto Mobile apps, the 'mobile editions' of the AppDepot, are not available for Android devices. However, mobile-compatible Presto apps can be viewed through the browser on Andriod devices. |
| Chrome | iPhone or iPad devices with iOS 4.3.3 and above |
| Firefox 24.0 | |
| Safari 7 | |

## Presto and MashZone Database Compatibility

Presto has an internal database, the Presto Repository. The Integrated MashZone Server for Presto also has an internal database. Both of these repositories are installed in separate Derby databases that are configured as *embedded* databases running in the same JVM as Presto and the Integrated MashZone Server.

**Important:** The Presto Repository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

These repositories are compatible with the following databases and drivers:

| Database / Driver | Presto | MashZone |
|---|---|---|
| Apache Derby 10.5.3 (embedded)<br>derby-embedded-10.5.3.jar | ✓ | ✓ |
| Microsoft SQL Server 2008 (10.00.2531)<br>jtds-1.3.1.jar | ✓ | ✓ |
| Microsoft SQL Server 2012 (11.0.2012)<br>jtds-1.3.1.jar | | ✓ |
| MySQL 5<br>mysql-connector-java-5.1.6-bin.jar | ✓ | |
| MySQL 5.6.1<br>mysql-connector-java-5.1.6-bin.jar | ✓ | ✓ |
| Oracle 11g (11.1.0.6)<br>ojdbc16.jar<br>**Note:** Scripts are compatible with the Oracle SQL Plus client. | ✓ | ✓ |

| Database / Driver | Presto | MashZone |
|---|---|---|
| Oracle 12c (12.1.0.6)<br>ojdbc16.jar<br>**Note:** Scripts are compatible with the Oracle SQL Plus client. | ✓ | ✓ |
| PostgreSQL 9.0<br>postgresql-9.0-801.jdbc3.jar | ✓ | |
| PostgreSQL 9.3.1<br>postgresql-9.2-1002.jdbc4.jar | | ✓ |

Both Presto and MashZone also work with databases making database assets visible in Presto as mashable information sources or MashZone feeds. Database mashables have been tested with:

- DB2 9.7.400.501 (db2jcc.jar)
- Microsoft SQL Server 2008 (jtds-1.2.5.jar)
- MySQL 5.x (mysql-connector-java-5.1.6-bin.jar)
- Oracle 11g (ojdbc14.jar)
- PostgreSQL 9 (postgresql-9.0-801.jdbc3.jar)

## What is Installed with Presto

Presto initially installs these WAR files:

| WAR | Server and/or Application |
|---|---|
| presto.war | Presto Server, Presto Hub and AppDepot |
| mashzone.war | Integrated MashZone Server |
| rtbs.war | Event Service |
| ibo.war and ibo-config.war | IBO user interface |

Presto also installs the following additional software:

- Apache's TomEE Plus Servlet Container, version 7.0.47
- Derby Database, version 10.5.3.0.

> **Important:** The Presto Repository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution. See "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 for details.

Finally, Presto includes installation packages for built-in apps and dashboards that you can choose to import to make them available to users. This can include:

- The PPM Chart custom app to allow users to easily pull charts from PPM into workspace apps (dashboards).
- Workspace apps (dashboards) for business process monitoring from Optimize.

## Presto Installation Folders

```
SoftwareAG
- Presto
  - apache-tomee-jaxrs (the container and web apps)
  - ibo-dashboards (for IBO + built-in dashboards)
  - mashzone (external resources for the Integrated Servers)
    - clientapps (Zip for PPM Chart app)
    - data
      - importexport
      - jdbcdrivers
      - resources
    - tools
  - prestocli (Presto 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 Presto Server

The Presto Hub, AppDepot, Presto Mobile apps and the Presto Add-Ons for Excel, SharePoint and Portals all depend on the Presto Server. If Presto is installed as part of Intelligent Business Operations, IBO dashboards also depend on the Presto Server.

The Presto Server depends on the Presto Repository. Start-up and shutdown for the Presto Server also automatically starts and stops the Integrated MashZone Server and the Event Service which depend on the MashZone Repository.

See "Start the Presto Server" on page 21 and "Stop the Presto Server" on page 22 for instructions.

## Start the Presto Server

1. If you have installed the Presto Add-On for SharePoint Add-On and the Presto Server uses the same LDAP Directory as SharePoint, log in to the Presto Server host using the correct startup user account. See "Startup Account for the Presto Server With Presto Add-On for SharePoint" on page 22 for more information.
2. If the Presto Repository and MashZone Repository have been moved from the default Derby database and they are *not* already running, manually start these databases following the instructions for their host database.
3. Do one of the following to start the Presto Server:
   - For Windows systems, either:
     - From the Start menu, select **Software AG > Start Servers > Start Presto *version***.
     - Enter this command in a command window:

       `c:>`*`presto-install`*`/apache-tomee-jaxrs/bin/catalina.bat start`

       **Note:** On Windows Server 2008 Presto Server needs to be started as Administrator. In order to run Presto Server as Administrator right click on the **Start Presto *version*** shortcut and select "Run as administrator".
   - For Linux, Mac OS X or UNIX systems, open a new terminal window and move to this folder:

     `% cd `*`presto-install`*`/apache-tomee-jaxrs/bin`

     Then enter this command:

     `% catalina.sh start`

   This also automatically starts the Integrated MashZone Server and the Event Service.

Open the Presto Hub at `http://`*`app-server:port`*`/presto` and log in. You can now access the AppDepot and all the Presto tools: Mashboard, Wires, the MashZone Feed Editor, the Mashup Editor, the App Editor and the Admin Console.

## Startup Account for the Presto Server With Presto Add-On for SharePoint

There is a known issue with authentication for SharePoint users working with Presto Add-On for SharePoint when both the Presto Server and SharePoint use the same LDAP Directory for authentication. When users work in Presto Hub to register SharePoint lists, use SharePoint lists or searches in mashups or publish apps to SharePoint, the credentials for the user account that started Presto Server is used initially for authentication and authorization rather than the credentials of the current user or the configured SharePoint connection.

To work around this issue, you must use a user account that has been defined specifically to start up the Presto Server in this context. You have two options to create this special startup account:

- Create a user account on the host for the Presto Server that does not exist in your LDAP Directory.
- Add a user to your LDAP Directory but do *not* grant this user permissions to work in SharePoint.

## Stop the Presto Server

1. Do one of the following:
   - For Windows systems, either:
     - From the Start menu, select **Software AG > Stop Servers > Stop Presto 3.7**.
     - Enter this command in a command window:

       ```
       c:>presto-install/apache-tomee-jaxrs/bin/catalina.bat stop
       ```
   - For Linux, Mac OS X or UNIX systems, open a new terminal window and move to this folder:

     ```
     % cd presto-install/apache-tomee-jaxrs/bin
     ```

     Then enter this command:

     ```
     % cataline.sh stop
     ```

   This also automatically stops the Integrated MashZone and Integrated Event Service servers.
2. If the Presto Repository and MashZone Repository have moved from the default Derby database, you can also choose to stop these two databases. See documentation for their host database for instructions.

## Startup Considerations

When the Presto Repository is hosted in a robust database solution, it must be started before the Presto Server for a successful startup. With the default Derby database, the Presto Repository runs as an *embedded* database that is automatically started with the Presto 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 Presto Server after the Presto Repository has been restarted.

If you host the Presto Repository in a MySQL or Oracle database, you may also be able to have the database start automatically with the host.

## Manage Licenses for Presto, Universal Messaging and BigMemory

Presto requires two licenses: one for Presto and one for the Integrated Universal Messaging that is deployed with Presto.

If you are using BigMemory features that require this, you also need to make your BigMemory license available to the Presto Server and/or the Integrated MashZone Server. See "BigMemory for Caching, Connections and Presto Analytics" on page 114 for features that require this additional license.

You can apply licenses when you install Presto, or you can install and use Presto without a license for a trial period of 30 days. If you purchase Presto after installation, you must manually apply the Presto and Universal Messaging licenses. If needed, you can also manually apply a BigMemory license.

**Note:** When Presto 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 Presto licenses, copy the PrestoLicense.xml file into the *presto-install* /apache-tomee-jaxrs/conf folder.

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

3. For the Integrated Universal Messaging license, copy the license.xml file to the *presto-install* /um folder.

> **Note:** If Presto is deployed in a cluster, copy the license file to this folder in every cluster member.

4. If a BigMemory license is required:
   a. Copy the license file terracotta.key to the *presto-install* /apache-tomee-jaxrs/conf folder.

      > **Note:** If Presto is deployed in a cluster, you must copy this file to every cluster member.

   b. Add this Java system property to the Presto startup script :

      `-Dcom.tc.productkey.path=`*`presto-install`*`/apache-tomee-jaxrs/conf/terracotta.key`

      Add this property to either:

      - *presto-install* /apache-tomee-jaxrs/bin/setenv.bat file for Windows systems, or
      - *presto-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX systems.

      > **Note:** If Presto is deployed in a cluster, you must update the startup scripts for every cluster member.

5. Restart the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

## Move the Presto and MashZone Repositories to a Robust Database Solution

The Presto Repository and MashZone Repository are initially installed in a Derby database suitable *only for trial*purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

You can host the Presto and MashZone Repositories in any database that Presto or MashZone supports. See "Presto and MashZone Database Compatibility" on page 18 in System Requirements for details.

You can move both of these repositories to one of the following databases:

- *Microsoft SQL Server*: see "Move Presto and MashZone Repositories to Microsoft SQL Server" on page 25 for instructions.
- *MySQL*: see "Move the Presto and MashZone Repositories to MySQL" on page 29 for instructions.
- *Oracle*: see "Move the Presto and MashZone Repositories to Oracle" on page 32 for instructions.

- *PostGres*: see "Move the Presto and MashZone Repositories to PostGres" on page 36 for instructions.

## Troubleshooting Connections to the Presto Repository

The most common problem when the Presto Server server does not restart successfully after you move the Presto Repository to a new database is that is cannot connect to the Presto Repository. To verify that this is the problem:

1. Open the Presto Server log file prestoserver.log in your web application server's log directory. For TomEE, this is:

   *web-apps-home* /logs/prestoserver.log

2. Check for a log entry for `Cannot create PoolableConnectionFactory` near the end of the file. This error indicates the Presto Server could not successfully connect to the new database.

Common causes for this error include:

- The database hosting the Presto Repository is not running.

  If this is true, start the Presto Repository and verify that it is up. Then restart the Presto Server and confirm that this starts successfully.

- There are one or more firewalls between the Presto Repository and the Presto Server that are not configured to allow this connection.

  **Note:** This can only happen when the database for the Presto Repository is hosted on a different server than the Presto Server.

  Update the firewall configuration to allow this connection. Then restart the Presto Server and confirm that this starts successfully.

- The URL or other connection configuration that you entered in Presto for the Presto Repository is incorrect.

  To correct an error in this case, edit the resource properties for the Presto Repository in the *presto-install* /apache-tomee-jaxrs/conf/tomee.xml file.

  Then restart the Presto Server and confirm that this starts successfully.

- Port or connection configuration for the database is not set up properly to allow connections from the Presto Server. See documentation for your database for more information.

## Move Presto and MashZone Repositories to Microsoft SQL Server

1. If you are using your LDAP Directory as the Presto User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for Presto *before* you move the Presto Repository. See "Grant User Access to Presto with Built-in Groups" on page 72 for instructions.

When the Presto User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the Presto 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 Presto or MashZone. See "Presto and MashZone Database Compatibility" on page 18 for details.

     **Note:** The JTDS driver for SQL Server is the recommended driver for use with Presto or MashZone and SQL Server databases due to known issues with the default SQL Server JDBC driver.

   - If you want Presto to support international characters in meta-data for artifacts, make sure the database uses the UTF-16 character encoding and collation. See documentation for your database for specific instructions.
   - It is a best practice to require passwords for every database account that can access the Presto Repository.

3. Start the database(s) that will become host to the Presto Repository and/or the MashZone Repository, if they are not already up.
4. Using the SQL tool for the database that will be host, add Presto Repository tables with the scripts shown below from the corresponding folder in *presto-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 Presto Repository tables, if needed.

5. Copy the JAR file for the JDBC driver for your database to the following folder for each Presto Server that uses this Presto Repository or MashZone Server that uses this MashZone Repository:

   *presto-install* /apache-tomee-jaxrs/lib

6. Replace the JAR for the Presto Repository:

   a. Remove the *web-apps-home* /presto/WEB-INF/lib/jackbe-presto-rds-postgresql-derby.jar JAR file for each Presto Server that uses this Presto 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 Presto.

   b. Copy this JAR file:

      *presto-install* /prestorepository/jackbe-presto-rds-oracle-mysql-mssql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.

7. Update snapshot scheduler configuration for the Presto Server:
   a. In the text editor of your choice, open the applicationContext-scheduler.xml file in the *webapps-home* /presto/WEB-INF/classes/ folder for the Presto 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 Presto Server in the cluster.
8. Open the *presto-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.
9. For the Presto Repository, edit the <Resource> element with an ID of `PrestoRepository` and:
   a. Update the JDBC driver, URL and credential properties:

```
<Resource id="PrestoRepository" type="DataSource">
  JdbcDriver net.sourceforge.jtds.jdbc.Driver
  JdbcUrl jdbc:jtds:sqlserver://host-name:port/database
  UserName username
  Password password
  JtaManaged = false
</Resource>
```

   The JTA managed property *must* be `false`.
   b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

   Some common properties you may need to set include:

   - `validationQuery = select 1`
   - Common tuning properties for connections pools. See "Tuning the Presto or MashZone Repository Connection Pool" on page 199.

10. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:
    a. Update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver net.sourceforge.jtds.jdbc.Driver
  JdbcUrl jdbc:jtds:sqlserver://host-name:port/database
  UserName username
  Password password
  defaultAutoCommit = true
  maxActive = 200
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
```

```
</Resource>
```

b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

Some common properties you may need to set include:

- `validationQuery = select 1`
- Common tuning properties for connections pools. See "Tuning the Presto or MashZone Repository Connection Pool" on page 199.

11. Save your changes to this file.

If the Presto Server does not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

12. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

a. Comment out the property definitions for the Derby database.

b. Uncomment the properties for MSSQL and update these properties to match the JNDI configuration your previously updated for TomEE.

c. Save your changes to this file.

You must keep JDBC and JNDI configuration synchronized for the Presto Repository. The application server is using JNDI to connect to the Presto Repository, but some components still use JDBC information.

13. Restart the Presto Server to apply these changes. This also restarts the MashZone Server.

If the Presto Server wedoes not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

14. Update connection information for the Snapshots Repository:

a. Open Presto 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 134 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 131 for information on specific configuration properties.

f. Click **Save**.

15. Restart the Presto Server to apply these changes.

16. Load macros required for the Snapshot feature in Presto:

a. Open a command or terminal window and move to the *presto-install* /presto-cli/bin folder.

b. Enter the appropriate command, shown below, for your operating system:

| For Windows | For Linux, OS/X or UNIX |
|---|---|
| `publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api` | `./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api` |

## Move the Presto and MashZone Repositories to MySQL

1. If you are using your LDAP Directory as the Presto User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for Presto *before* you move the Presto Repository. See "Grant User Access to Presto with Built-in Groups" on page 72 for instructions.

   When the Presto User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the Presto Repository or MashZone Repository in a new database, create the database following MySQL documentation. Keep the following points in mind:

   - Make sure this database is supported by Presto or MashZone. See "Presto and MashZone Database Compatibility" on page 18 for details.
   - If you want Presto 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 Presto Repository, you should increase the maximum allowed packet size, which defaults to 1MB, for the database.

3. Start the database(s) that will become host to the Presto Repository and/or the MashZone Repository, if they are not already up.

4. Using the SQL tool for the database that will be host, add Presto Repository tables with the scripts shown below from the corresponding folder in *presto-install* / prestorepository/mysql:

   a. Create a Database to hold the Presto 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 Presto Repository tables, if needed.

5. Replace the JAR for the Presto Repository:

   a. Remove the *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/lib/jackbe-presto-rds-postgresql-derby.jar JAR file for each Presto Server that uses this Presto 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 Presto.

   b. Copy this JAR file:

      *presto-install* /prestorepository/jackbe-presto-rds-oracle-mysql-mssql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.

6. Copy the MySQL JDBC driver jar file to *presto-install* /apache-tomee-jaxrs/lib.

7. Open the *presto-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.

8. For the Presto Repository, edit the <Resource> element with an ID of `PrestoRepository` and:

   a. Update the JDBC driver, URL and credential properties:

```
<Resource id="PrestoRepository" type="DataSource">
  JdbcDriver com.mysql.jdbc.Driver
  JdbcUrl jdbc:mysql://host-name/databasename
  UserName = username
  Password = password
  JtaManaged = false
</Resource>
```

      For MySQL databases, it is *recommended* that you include the database name in data source URLs. If this information is omitted, testing the data source fails and may also cause errors with access to stored procedures.

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

   b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

      Some common properties you may need to set include:

- `validationQuery = select 1 from dual`
- Common tuning properties for connections pools. See "Tuning the Presto or MashZone Repository Connection Pool" on page 199.

9. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:

   a. Update the following properties:

   ```
   <Resource id="amzDatabase" type="DataSource">
     JdbcDriver com.mysql.jdbc.Driver
     JdbcUrl jdbc:mysql://host-name:port/databasename
     UserName = username
     Password = password
     defaultAutoCommit = true
     maxActive = 200
     maxIdle = 20
     maxOpenPreparedStatements = 0
     maxWaitTime = 5000
   </Resource>
   ```

   For MySQL databases, it is *recommended* that you include the database name in data source URLs. If this information is omitted, testing the data source fails and may also cause errors with access to stored procedures.

   b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

   Some common properties you may need to set include:

   - `validationQuery = select 1`
   - Common tuning properties for connections pools. See "Tuning the Presto or MashZone Repository Connection Pool" on page 199.

10. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

    a. Comment out the property definitions for the Derby database.

    b. Uncomment the properties for MySQL and update these properties to match the JNDI configuration your previously updated for TomEE.

    c. Save your changes to this file.

    You must keep JDBC and JNDI configuration synchronized for the Presto Repository. The application server is using JNDI to connect to the Presto Repository, but some components still use JDBC information.

11. Start the Presto Server to apply these changes. This also starts the MashZone Server.

    If the Presto Server wedoes not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

12. Update connection information for the Snapshots Repository:

    a. Open Presto 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 134 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 131 for information on specific configuration properties.

f. Click **Save**.

13. Restart the Presto Server to apply these changes.

14. Load macros required for the Snapshot feature in Presto:

a. Open a command or terminal window and move to the *presto-install* /presto-cli/bin folder.

b. Enter the appropriate command, shown below, for your operating system:

| For Windows | For Linux, OS/X or UNIX |
|---|---|
| `publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api` | `./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api` |

## Move the Presto and MashZone Repositories to Oracle

1. If you are using your LDAP Directory as the Presto User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for Presto *before* you move the Presto Repository. See "Grant User Access to Presto with Built-in Groups" on page 72 for instructions.

   When the Presto User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the Presto Repository or MashZone Repository in a new database, create the database following Oracle documentation.

   - Make sure this database is supported by Presto or MashZone. See "Presto and MashZone Database Compatibility" on page 18 for details.
   - If you want Presto 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 Presto Repository.

3. Start the database(s) that will become host to the Presto Repository and/or the MashZone Repository, if they are not already up.
4. Using the SQL tool for the database that will be host, add Presto Repository tables with the scripts shown below from the corresponding folder in *presto-install* /prestorepository/oracledb:
   - createDBTables.txt for MetaData and the default User Repository
   - createSnapsTables.sql for Snapshots
   - createSchedulerTables.sql for Scheduler

   **Note:** This folder contains other scripts to drop the corresponding Presto Repository tables.
5. Replace the JAR for the Presto Repository:
   a. Remove the *web-apps-home* /presto/WEB-INF/lib/jackbe-presto-rds-postgresql-derby.jar JAR file for each Presto Server that uses this Presto 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 Presto.
   b. Copy this JAR file:

      *presto-install* /prestorepository/jackbe-presto-rds-oracle-mysql-mssql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.
6. Copy the JAR file for the JDBC driver for your database to the following folder for each Presto Server that uses this Presto Repository or MashZone Server that uses this MashZone Repository:

   *presto-install* /apache-tomee-jaxrs/lib
7. Add the Quartz-Oracle JAR to the Presto Server:
   a. Download the Quartz 1.7.3 installation ZIP from http://www.quartz-scheduler.org/download/download-catalog.html.
   b. Extract the Quartz 1.7.3 installation ZIP.
   c. Copy the quartz-oracle-1.7.3.jar file to the *presto-install* /apache-tomee-jaxrs/lib folder.
   d. If this is a clustered environment, copy this JAR file to each Presto Server in the cluster.
8. Update snapshot scheduler configuration for the Presto Server:
   a. In the text editor of your choice, open the applicationContext-scheduler.xml file in the *webapps-home* /presto/WEB-INF/classes/ folder for the Presto 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.or
      <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 Presto Server in the cluster.

9. Open the *presto-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.

10. For the Presto Repository, edit the <Resource> element with an ID of `PrestoRepository` and:

a. Update the JDBC driver, URL and credential properties:

```
<Resource id="PrestoRepository" type="DataSource">
  JdbcDriver oracle.jdbc.OracleDriver
  JdbcUrl jdbc:oracle:drivertype:@host-name:port:dbname
  UserName username
  Password password
  JtaManaged = false
</Resource>
```

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

b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

Some common properties you may need to set include:

- `validationQuery = select 1 from dual`
- Common tuning properties for connections pools. See "Tuning the Presto or MashZone Repository Connection Pool" on page 199.

11. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:

a. For Oracle 11g, update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver oracle.jdbc.OracleDriver
  JdbcUrl jdbc:oracle:thin//:host-name:port/sid
  UserName username
  Password password
  defaultAutoCommit = true
  rollbackOnReturn = true
  removeAbandoned = true
  logAbandoned = true
  removedAbandonedTimeout = 5
  maxActive = 200
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
</Resource>
```

b. For Oracle 12c, update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver oracle.jdbc.OracleDriver
  #Oracle 12 url
  JdbcUrl jdbc:oracle:drivertype:@host-name:port:dbname
  UserName username
  Password password
  defaultAutoCommit = true
  rollbackOnReturn = true
  removeAbandoned = true
  logAbandoned = true
  removedAbandonedTimeout = 5
  maxActive = 100
  maxIdle = 20
  maxOpenPreparedStatements = 0
  maxWaitTime = 5000
</Resource>
```

12. Save your changes to this file.

    If the Presto Server does not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

13. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

    a. Comment out the property definitions for the Derby database.

    b. Uncomment the properties for Oracle and update these properties to match the JNDI configuration your previously updated for TomEE.

    c. Save your changes to this file.

    You must keep JDBC and JNDI configuration synchronized for the Presto Repository. The application server is using JNDI to connect to the Presto Repository, but some components still use JDBC information.

14. Restart the Presto Server to apply these changes. This also restarts the MashZone Server.

    If the Presto Server wedoes not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

15. Update connection information for the Snapshots Repository:

a. Open Presto 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 134 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 131 for information on specific configuration properties.

f. Click **Save**.

16. Restart the Presto Server to apply these changes.

17. Load macros required for the Snapshot feature in Presto:

a. Open a command or terminal window and move to the *presto-install* /presto-cli/bin folder.

b. Enter the appropriate command, shown below, for your operating system:

| For Windows | For Linux, OS/X or UNIX |
|---|---|
| `publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api` | `./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api` |

## Move the Presto and MashZone Repositories to PostGres

1. If you are using your LDAP Directory as the Presto User Repository, make sure that at least one user in your LDAP Directory has administrator privileges for Presto *before* you move the Presto Repository. See "Grant User Access to Presto with Built-in Groups" on page 72 for instructions.

   When the Presto User Repository is your LDAP Directory, the default administrator account (`Administrator` user) is disabled.

2. If you are hosting the Presto 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 Presto or MashZone. See "Presto and MashZone Database Compatibility" on page 18 for details.
   - If you want Presto 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 Presto Repository.
- When you initialize the PostGres database that will host the Presto Repository or MashZone Repository, you may need to specifically set the locale used by the database to ensure case-insensitive sorting.

3. Start the database(s) that will become host to the Presto Repository and/or the MashZone Repository, if they are not already up.
4. Using the SQL tool for the database that will be host, add Presto Repository tables with the scripts shown below from the corresponding folder in *presto-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 Presto Repository tables in these folders, if needed.
5. Update snapshot scheduler configuration for the Presto Server:
   a. In the text editor of your choice, open the applicationContext-scheduler.xml file in the *webapps-home* /presto/WEB-INF/classes/ folder for the Presto Server.
   b. Find the `org.springframework.scheduling.quartz.SchedulerFactoryBean` bean.
   c. Update the `org.quartz.jobStore.driverDelegateClass` property to the `org.quartz.impl.jdbcjobstore.PostgreSQLDelegate` delegate.

      The configuration would now look like:

```
...
<bean id="scheduler"
>
  <property name="applicationContextSchedulerContextKey">
    <value>applicationContext</value>
  </property>
  <property name="quartzProperties">
    <props>
      <prop key="org.quartz.scheduler.instanceId">AUTO</prop>
      <prop key="org.quartz.jobStore.class"> org.quartz.impl.jdbcjobstore.JobStoreTX</prop
      <prop key="org.quartz.jobStore.tablePrefix">QRTZ_</prop>
      <prop key="org.quartz.jobStore.driverDelegateClass"> org.quartz.impl.jdbcjobstore.Po
      <prop key="org.quartz.jobStore.dataSource">schedulerDS</prop>
      ...
    </props>
  </property>
</bean>
...
```

   d. Save this change.
   e. If this is a clustered environment, copy the updated applicationContext-scheduler.xml configuration file to each Presto Server in the cluster.

6. Copy the JAR file for the JDBC driver for your database to the following folder for each Presto Server that uses this Presto Repository or MashZone Server that uses this MashZone Repository: *presto-install* /apache-tomee-jaxrs/lib.
7. Open rdsApplicationContext.xml under *presto-install* /apache-tomee-jaxrs/classes and add the following keys to:

```
<property name="jdoProperties">
...
<map>
...
<entry key="javax.jdo.mapping.Schema" value="public"/>
<entry key="datanucleus.identifier.case" value="LowerCase"/>
...
</map>
</property>
```

8. If you are using PostgreSQL version 9.x please open postgresql.conf under PostgreSQL-install/9.x/data and un-comment the following property and make sure it is set to **off:**`standard_conforming_strings = off`.
9. Open the *presto-install* /apache-tomee-jaxrs/conf/tomee.xml configuration file in the text editor of your choice.
10. For the Presto Repository, edit the <Resource> element with an ID of `PrestoRepository` and:
    a. Update the JDBC driver, URL and credential properties:

```
<Resource id="PrestoRepository" type="DataSource">
  JdbcDriver org.Postgresql.Driver
  JdbcUrl jdbc:postgresql://host-name:port/databasename
  UserName username
  Password password
  JtaManaged = false
</Resource>
```

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

    b. If needed, update optional properties. See TomEE Datasource Properties for a complete list of optional properties and information on defaults.

    Some common properties you may need to set include:

    - `validationQuery = select 1`
    - Common tuning properties for connections pools. See "Tuning the Presto or MashZone Repository Connection Pool" on page 199.

11. For the MashZone Repository, edit the <Resource> element with an ID of `amzDatabase` and:
    a. Update the following properties:

```
<Resource id="amzDatabase" type="DataSource">
  JdbcDriver org.Postgresql.Driver
  JdbcUrl jdbc:postgresql//:host-name:port/databasename
  UserName username
  Password password
  defaultAutoCommit = true
  maxActive = 200
```

```
    maxIdle = 20
    maxOpenPreparedStatements = 0
    maxWaitTime = 5000
</Resource>
```

12. Save your changes to this file.

    If the Presto Server does not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

13. Open the rdsJDBC.properties configuration file, from the *web-apps-home* /presto/WEB-INF/classes folder, in the text editor of your choice and:

    a. Comment out the property definitions for the Derby database.

    b. Uncomment the properties for PostGres and update these properties to match the JNDI configuration your previously updated for TomEE.

    c. Save your changes to this file.

    You must keep JDBC and JNDI configuration synchronized for the Presto Repository. The application server is using JNDI to connect to the Presto Repository, but some components still use JDBC information.

14. Restart the Presto Server to apply these changes. This also restarts the MashZone Server.

    If the Presto Server wedoes not start up successfully, see "Troubleshooting Connections to the Presto Repository" on page 25 for suggestions.

15. Update connection information for the Snapshots Repository:

    a. Open Presto 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 134 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 131 for information on specific configuration properties.

    f. Click **Save**.

16. Restart the Presto Server to apply these changes.

17. Load macros required for the Snapshot feature in Presto:

    a. Open a command or terminal window and move to the *presto-install* /presto-cli/bin folder.

    b. Enter the appropriate command, shown below, for your operating system:

| For Windows | For Linux, OS/X or UNIX |
|---|---|
| `publish-global-macros.bat -u Administrator -p manage -url http://app-server:port/presto/edge/api` | `./publish-global-macros -u Administrator -p manage -url http://app-server:port/presto/edge/api` |

## Integrate Your LDAP Directory with Presto

In many cases, users and authentication information for an organization is defined in an existing LDAP Directory. You can configure Presto 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 Presto support for specific LDAP Directory solutions.

To configure your LDAP Directory as the Presto User Repository:

1. If the Presto Server is not yet started, start Presto. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Change Presto configuration to use LDAP as the authentication provider.
   a. Edit the userRepositoryApplicationContext.xml file in the *presto-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 Presto Configuration Folder" on page 258 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 Presto configuration for the user attribute provider.

a. If it is not already open, edit the userRepositoryApplicationContext.xml file in the *presto-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 Presto Configuration Folder" on page 258 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 Presto Server until all other LDAP configuration steps have been completed.

4. Define configuration in the Admin Console in Presto Hub for:

- Connections to the LDAP Directory. See "Defining LDAP Connection Configuration" on page 42.
- Authentication mechanisms. See "Defining the Authentication Scheme" on page 42.
- Authorization mechanisms. See "Defining the Authorization Scheme" on page 43.
- All user and group queries used in Presto applications. See "Enabling Presto Application Queries for All LDAP Users or Groups for Permissions" on page 45.

See also "Expose User Attributes from the User Repository in Presto" on page 139 for information on making LDAP user attributes accessible as Presto user attributes.

5. Add the built-in Presto user groups to LDAP as new groups and assign at least one user as a Presto administrator.

See "Grant User Access to Presto with Built-in Groups" on page 72 for instructions.

6. Restart the Presto Server.

   Presto now uses LDAP as the user repository. You can now login using the user account assigned in earlier steps as a Presto administrator.

   To grant access to other users, add them to an appropriate built-in Presto user group in LDAP. See "Grant User Access to Presto with Built-in Groups" on page 72 for instructions.

## Defining LDAP Connection Configuration

1. If needed, log into Presto Hub and click Admin Console in the main menu.
2. Expand **Presto 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:

     ```
     cn=Directory Manager
     ```

   - *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 42, "Defining the Authorization Scheme" on page 43 and "Enabling Presto Application Queries for All LDAP Users or Groups for Permissions" on page 45) 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 Presto Hub and click Admin Console in the main menu.
2. Expand **Presto 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=People
     ```

- *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 Presto 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

Presto permissions are assigned to user groups or to individual users. To set up authorization when LDAP is the user repository, you must relate Presto 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, Presto 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 Presto, please contact your Software AG representative for more information.

You *must* add the built-in Presto groups that define basic permissions as groups in LDAP. You assign users to these built-in groups to assign basic Presto permissions. Your existing LDAP groups can then be used in Presto to define run permissions for specific mashables, mashups or apps. For more information on authorization, see "Authorization Policies and Permissions" on page 71.

1. If needed, log into Presto Hub and click  Admin Console in the main menu.
2. Expand **Presto 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 Presto 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 Presto 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 Presto permissions. The default value is the group common name:

     ```
     cn
     ```

     **Important:** If you change this property, you *must* also update the **Group Name Pattern** proprty.

   - 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 Presto groups.

     For example:

     ```
     cn(PRESTO_.*?)
     ```

     Presto 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 Presto 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 Presto groups.

  For example:

  ```
  cn(PRESTO_.*?)
  ```

  Presto 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 Presto to find the expected values for built-in groups, but use the full correct group names for the groups for your organization.

With these properties set, for example:

```
Search Groups for User Membership = true
Group Search Base=ou=groups
User Group Search Filter=uniquemember={0}
Group Name Attribute = cn
```

And a username of `jwalker`, Presto 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 Presto groups and to groups with run permissions for artifacts to determine the full set of permissions for `jwalker`.

## Enabling Presto Application Queries for All LDAP Users or Groups for Permissions

Presto queries the User Repository for user groups and users to enable you and other users to assign permissions for Presto resources. To enable these queries you set properties in the Admin Console:

1. If needed, log into Presto Hub and click  Admin Console in the main menu.
2. Expand **Presto 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 Presto Queries) = the search filter, combined with User Search Base that is used to find all user entries. For example:

  ```
  objectclass=inetOrgPerson
  ```

To support wildcard searches and define the sort order for results, you must also define these properties:

- **Attributes Used in Wildcard Search** (in Presto 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 Presto 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 Presto Queries) = the LDAP attribute that holds users' first names.
- *User Last Name Attribute* (in Presto Queries) = the LDAP attribute that holds users' last names.
- *User Email Attribute* (in Presto Queries) = the LDAP attribute that holds users' email addresses.

5. To enable queries for LDAP groups that can be used to assign Presto 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 Presto permissions. For example:

     ```
     ou=groups
     ```

     **Important:** This property is also used to search for Presto 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 Presto permissions. For example:

     ```
     objectclass=groupOfUniqueNames
     ```

# Use the Default Presto User Repository

Presto authenticates users against a repository and retrieve authorization from the repository. This can be either a database or an LDAP Directory.

Presto is installed with a default user repository within the Presto Repository. You can use this initially as you explore Presto 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 Presto Repository and MashZone Repository are initially installed in a Derby database suitable *only for trial* purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

No configuration is required to use the default user repository. If you use this default, you manage user and group information using functions in the Admin Console. See these topics for more information:

- "Manage Users" on page 47
- "Manage User Groups" on page 49
- "Automatically Assign New Users to Groups" on page 50

## Manage Users

If you are using the default Presto User Repository, Presto 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 47 and "Edit, Grant Permissions and other User Management Tasks" on page 48 for instructions.

If you have configured Presto to use your LDAP Directory as the User Repository, you manage users and groups and assign permissions in LDAP.

### Create Users

1. If needed, open the Admin Console (click  in the Presto Hub main menu).
2. Expand the **Users and Groups** tab and click **Users**.
3. Click **Add new user** and complete the following information:
   - **User Name** must be unique. This is the end-user's login name and primary key for the user record. User names cannot exceed 256 characters.
   - **First Name** and **Last Name** are optional.
   - **Email**
   - **Password** cannot exceed 50 characters. Confirm the password.

> **Note:** Valid characters for the User Name and Password fields are defined by the database you use for the default Presto 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 Presto 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 Presto Hub to create artifacts or simply to work with artifacts from other users, you must add them to other groups, either manually or automatically. For more information, see:

- "Automatically Assign New Users to Groups" on page 50
- "Edit, Grant Permissions and other User Management Tasks" on page 48

## Edit, Grant Permissions and other User Management Tasks

1. If needed, open the Admin Console (click  in the Presto Hub main menu).
2. Expand the **Users and Groups** tab and click **Users**.
3. Find the user in the list using Search or scrolling. Management options include:
   - Click r a new password.
   - Click ıser'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 Presto User Groups and Permissions" on page 73 for information on the permissions for the built-in user groups in Presto. 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 Presto.

## Manage User Groups

Groups are used to grant permissions to users for specific actions in Presto. If you are using the default Presto User Repository, Presto administrators add and manage user groups in the Admin Console. If you have configured Presto 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 Prestogroups* have a set of permissions predefined that allow users to work in the Presto 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 Presto User Groups and Permissions" on page 73 for details.

**Note:** Presto administrators and artifact owners automatically have permission to run all artifacts or the artifacts they own repsectively.

*User-defined groups* are groups that administrators add to Presto. These groups are used to grant run permissions to group members for the specific artifacts that the group has been added to. Presto 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 75 for information.

1. If needed, open the Admin Console (click  in the Presto Hub main menu).
2. Expand the **Users and Groups** tab and click **Groups**.

   A list of groups displays. You can filter this list by entering part of a group name and clicking **Search**.
3. Click **Add new user group** and:
   a. Enter a unique name in the **Group** field.
   b. Click **Add Group**.
   c. Add more groups or click **Done** to close this form.
4. To delete a user group, click  **Delete** and confirm this.

**Important:** Do *not* delete any of the built-in groups for Presto: `Presto_Administrator`, `Presto_Developer`, `Presto_PowerUser`, `Presto_AuthenticatedUser` or `Presto_Guest`.

## Automatically Assign New Users to Groups

If you are using the Default Presto User Repository, you can automatically assign new users to groups when you add them to Presto. 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 Presto User Repository.

You can add both built-in Presto 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 Presto 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 Presto.

**To assign new users to groups**

1. If needed, open the Admin Console (click  in the Presto Hub main menu).
2. Expand the **Users and Groups** tab and click **Default Groups**.
3. Enter part of a group name and click **Search**.

   A list of groups that start with that name displays.
4. Drag a group and drop it in the **Default User Groups** bucket.

   Continue to find and add groups as needed.
5. Click **Save these changes**.

New user will automatically be assigned to all groups in the default list. This change does *not* affect any existing users.

# What's Next

When Presto is installed as an integral part of IBO, there are several optional tasks you may need to complete after the initial installation to support business intelligence dashboards for users. This includes:

- Configuration for Process Performance Manager (PPM) Integration to allow users to use PPM charts or data in workspace apps.
- Configuration for Presto connections and subscriptions to business process events from the EDA. See "Event Service Configuration and Administration" on page 159 for links to instructions.

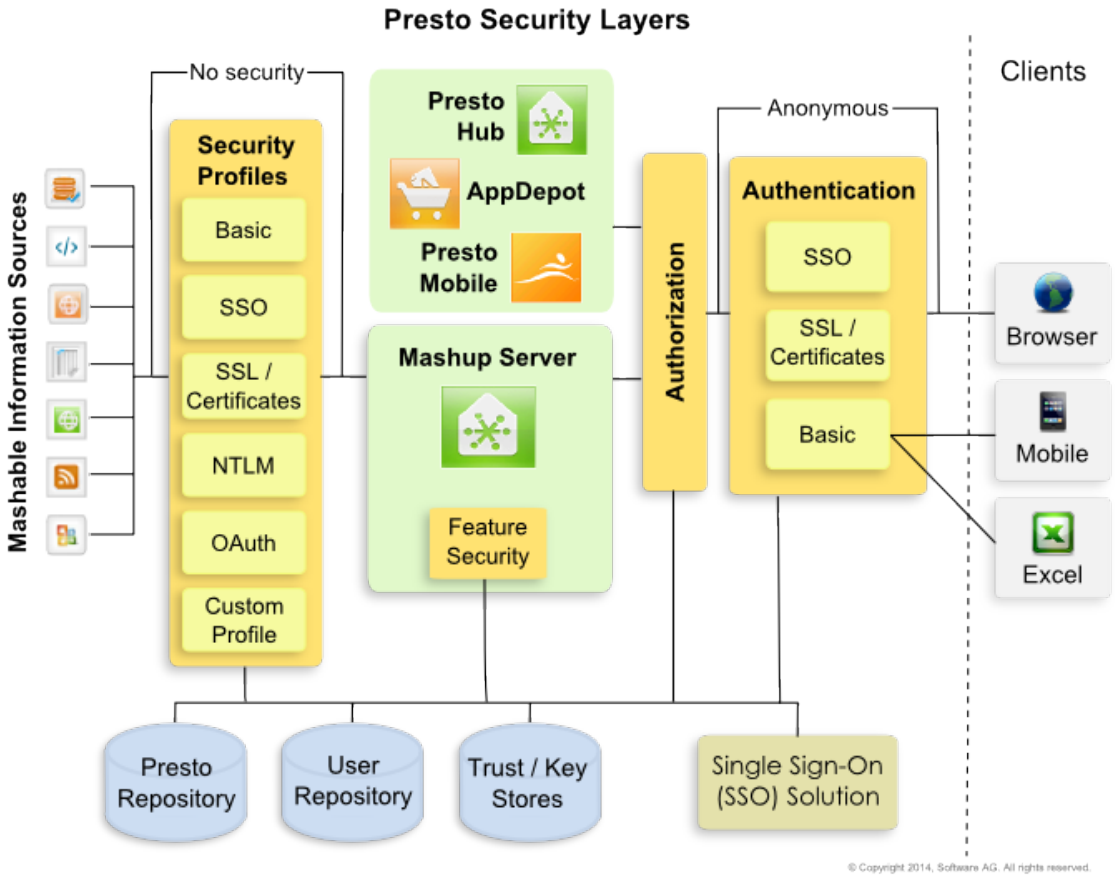
- Configuration for Presto connections and subscriptions to Apama events. See "Event Service Configuration and Administration" on page 159 for links to instructions.
- Configuration to enable business process dashboards. See the *Working with Business Process Dashboards* guide.

Other common configuration and administration tasks for Presto include:

- The full set of Presto Server Configuration includes:
  - Common configuration such as memory configuration, logging options, proxy server configuration or caching configuration.
  - A wide range of environment or business-specific configuration such as connecting to SharePoint for the Presto Add-On for SharePoint or managing provide and category taxonomies.
- Configuration for authentication and authorization. See "Presto Security" on page 53 for instructions.
- Other Presto and MashZone Repositories tasks.
- Administration tasks are discussed in Presto Server Administration for:
  - Viewing and managing server logs.
  - Managing resource files.
  - Migrating to new Presto releases.
  - Deploying Presto instances and artifacts.
  - Clustering Presto Servers.

# 3 Presto Security

Presto provides control of user interactions to register or create mashable information sources, mashups and apps and secure access for all users to work with these artifacts based on policies that you define. Security is embodied in these layers:




- Change password: For reasons of security we strongly recommend that the Presto administrator should change the standard Presto password after installation. See "Change technical user password" on page 55.
- **User Authentication**: based on the protocols shown above. You can also allow anonymous access if needed. See "Authentication and Guest Access" on page 56 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 Presto Hub and the Presto Enterprise AppDepot. See "Authorization Policies and Permissions" on page 71 for details.
- **Security Profiles**: that define the requirements for secure communication with mashable information sources.

  Presto supports the well-known protocols shown above. Presto developers can also create custom security profiles to support mashable information sources with unique requirements.

- **Feature Security**: to control any features in the Presto 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 141 and "Configure the Default Operations Generated for Database Mashable" on page 134 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 Presto 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 Presto 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.

<*presto-install*> /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/mz.properties

<*presto-install*> /apache-tomee-jaxrs/webapps/mashzone/WEB-INF/mashzone.properties

**Note:** This procedure is only required for Presto 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.

      ```
      $ <presto-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.

      ```
      $ <presto-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

Presto accepts requests from *both* unauthenticated (guests) and authenticated users. Guests, however, can only access mashables, mashup and apps in Presto that explicitly allow guest access.

Authentication is required:

- To access the Presto Hub, the AppDepot or the Presto Mobile apps for mobile phones or mobile tablets. Access to specific mashables, mashups, apps and specific features in Presto Hub or the AppDepot is determined by the user's permissions.
- To use any feature in any Presto Add-On that accesses the Presto Server, unless that Add-On also supports guest access.

  For example, portal users may view Presto apps or mashups using the Presto 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 Presto apps or mashups.
- To use apps or other web applications outside of Presto Hub, the AppDepot or the Presto 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 Presto session cookie. Sessions that have timed out are rejected with an appropriate error. See "Sessions and Timeouts" on page 58 for more information.
- Valid credentials. See "Valid Credentials" on page 57 for more information.
- Guest access header or parameter information. See "Enabling Guest Access" on page 58 for more information.

## User Authentication

Presto is initially installed with a set of "Default User Accounts" on page 58 that you can use to get started. You configure Presto to work with your LDAP Directory or you can continue to use the Default User Repository and simply add users and user groups to Presto. See "Use the Default Presto User Repository" on page 47, "Manage Users" on page 47 and "Manage User Groups" on page 49 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 66 for configuration instructions.

- A configurable Single Sign-On solution

  See "Authentication with Single Sign-On Solutions" on page 59 for configuration instructions.

Permission to work with apps and other Presto artifacts can also be granted to guests (unauthenticated users), if needed.

## Valid Credentials

When authentication is required, requests must have a valid Presto 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. Presto uses certificates, tokens or tickets to obtain the user's identity.

Presto supports the following mechanisms to obtain user credentials or user IDs:

- Basic authentication using username and passwords. This is authenticated against the Presto User Repository which may be a database or your LDAP Directory. See "Use the Default Presto User Repository" on page 47 for more information.

  **Note:** This is the *only* mechanism for obtaining user credentials that is supported by the Presto Mobile apps.

- SSL and Certificate authentication where the user identifier in certificate information is configurable. This is authenticated against the Presto User Repository which may be a database or your LDAP Directory, *unless* Dynamic User Support is enabled. See "Use the Default Presto User Repository" on page 47 and "Authentication with Digital Certificates/SSL" on page 66 for more information.

- Single Sign-On (SSO) solutions which are configurable. With SSO enabled, Presto delegates authentication to the SSO solution. Typically, configuration identifies

an SSO token, ticket or cookie that Presto uses to verify authentication with the SSO solution and to obtain the user ID. See "Authentication with Single Sign-On Solutions" on page 59 for more information.

If an authenticated request has no Presto session, Presto starts a new session and generates a Presto session cookie. See "Sessions and Timeouts" on page 58 for more information.

## Sessions and Timeouts

Presto is based on the standard J2EE session mechanism supported by your application server. Presto maintains a separate HTTP session for each authenticated user that has a unique session cookie. Each request with a valid Presto 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 Presto. SSO sessions and Presto session are separate.

The default session timeout for Presto is 30 minutes, defined in the *web-apps-home* /presto/web.xml configuration file. In general, HTTP session timeouts can be configured in web.xml, unless the application server provides other configuration mechanisms. Please see your application server documentation for additional information on session configuration.

## Enabling Guest Access

To allow unauthenticated users to work with apps or other artifacts outside of Presto, 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 `Presto_Guest` user group in the run permissions for those artifacts.

# Default User Accounts

Presto has four default user accounts that you can use 'out-of-the-box' to access Presto Hub, the AppDepot and the Presto Modile apps. These default users also illustrate the basic permissions to features in Presto. See "Built-In Presto User Groups and Permissions" on page 73 for more information on permissions.

| Username | Password | Built-in Group / Permissions | Description |
|---|---|---|---|
| `Administrator` | `manage` | `Presto_Administrator` | A Presto 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 Presto User Repository. |

If you configure Presto to use your LDAP Directory as the Presto 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 Presto administrator permissions.

## Authentication with Single Sign-On Solutions

With a single sign-on (SSO) solution, Presto delegates authentication to the SSO layer. Presto has the following pre-configured options to integrate with SSO solutions:

- SharePoint and the Presto Add-On for SharePoint.

  **Important:** Single sign-on authentication with SharePoint and Presto Add-On for SharePoint is enabled by default in Presto. No additional configuration is needed in Presto to integrate this token-based SSO solution.

- Agent-based SSO solutions, such as Netegrity SiteMinder. See "Configuration for Agent-Based SSO Solutions" on page 60 for instructions.

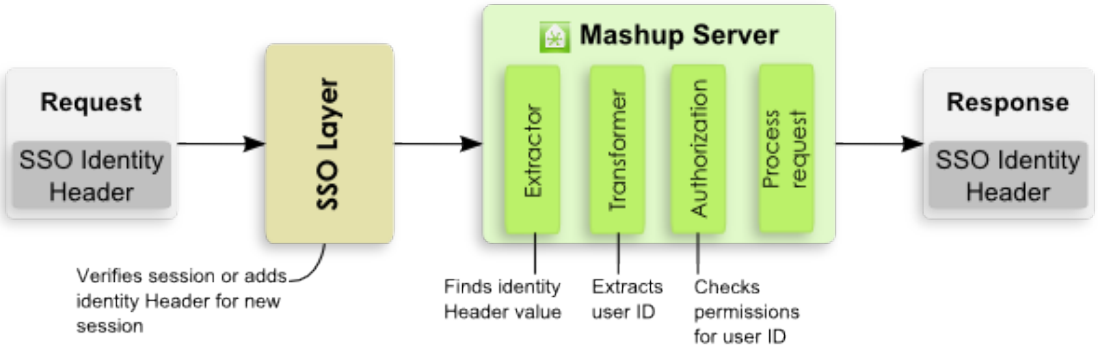
  **Note:** The Presto Add-On for SharePoint is not fully compatible with agent-based SSO solutions. Functionality in Presto Hub is available with an agent-based SSO solution. However, known issues limit the functionality available in SharePoint in this case.

- 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 Presto 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 62 for instructions.

## Configuration for Agent-Based SSO Solutions

With agent-based SSO, the basic flow of authentication and user identity information looks something like this:




Presto 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. Presto 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. Presto then uses the user ID to perform authorization and process the request.

To configure Presto 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. Presto provides a default extractor that looks for an HTTP header or parameter by name. Presto 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 Presto with your SSO solution. See "Implementing a Custom SSO Filter" on page 65 for details.

1. If needed, configure the Presto User Repository. See "Use the Default Presto User Repository" on page 47 for more information.

   In previous releases, Presto only supported SSO solutions with LDAP as the Presto User Repository. This restriction no longer applies.

2. Change the SSO filter in the applicationContext-security.xml configuration file for the Presto Server:

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

      This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.

b. Comment out the SSO filter bean (`<bean id="ssoProcessingFilter">`) for SharePoint (`class="com.jackbe.jbp.sas.security.ui.sso.sp.SharepointSSOFilter">`).

   For example:

```
<!-- <bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    ...
</bean> -->
```

c. Uncomment the SSO filter bean (`<bean id="ssoProcessingFilter">`) for agent-based solutions (`class="com.jackbe.jbp.sas.security.ui.sso.SSOPreAuthenticatedFilter"`).

   For example:

```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    ...
</bean>
```

3. In the agent-based SSO filter bean, configure the `principalExtractor` property:

   - The default extractor uses a bean with the `HttpHeaderOrParamTokenExtractor` class.

```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    <property name="principalExtractor">
      <bean
>
        <property name="httpHeaderName" value="SM_USER"/>
      </bean>
    </property>
    ...
</bean>
```

     Change the value of the `httpHeaderName` property for this extractor bean to the name of the HTTP header or parameter that contains user identify information from your SSO solution.

   - If you have a custom extractor class, replace the default extractor bean with configuration for your custom class.

4. In the agent-based SSO filter bean, configure the `principalTransformer` property:

   - The default transformer property uses a bean with the `RegexExtractionStringTransformation` class. This uses a regular expression to extract some portion of the value for the SSO header or parameter to get the final user ID that Presto 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:

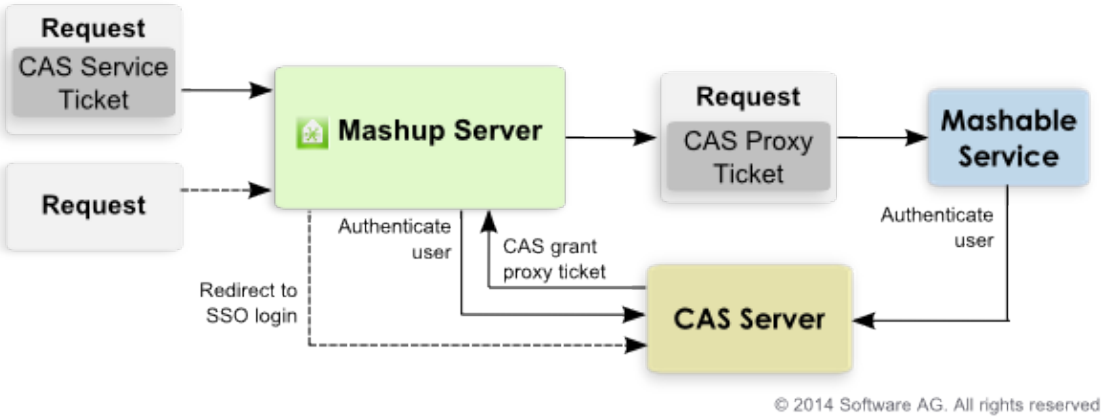
```
<bean id="ssoProcessingFilter"
>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="continueFilterChainOnUnsuccessfulAuthentication"
      value="true" />
    <property name="principalExtractor">
      <bean
>
        <property name="httpHeaderName" value="SM_USER"/>
      </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 Presto Server. See "Start and Stop the Presto Server" on page 20 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 Presto, if they have logged in with CAS, the request includes a service ticket unique to that user. Presto validates this ticket to retrieve user ID information needed for authorization.

Presto 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 Presto to send the mashable a proxy ticket that the mashable can use to authenticate the user.

If users access Presto without first logging in with CAS, Presto redirects users to the login page for CAS instead of the default Presto login page.

To configure Presto authentication using CAS, handle login redirects and enable CAS security profiles for mashables:

1. Enable HTTPS for communication between the Presto Server and the CAS Server. You must:

   a. Configure the application server hosting the Presto 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 101 for instructions for Tomcat. If Presto is deployed in another application server, see documentation for your application server for more information.

   b. Obtain a certificate for the Presto 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 98 for more information.

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

   This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.

3. Make sure that the import statement for applicationContext-security-authn-cas2.xml is uncommented.

   For example:

```
...
 <import resource="applicationContext-security-authn-rememberme.xml"/>
 <import resource="applicationContext-scheduler.xml"/>
 <!-- import resource="applicationContext-security-authn-x509.xml"/-->
 <!-- import resource="applicationContext-security-authn-rsa.xml"/-->
<import resource="applicationContext-security-authn-cas2.xml"/>
...
```

4. Find the bean with `authenticationEntryPointFilter` ID and change the value of the `defaultAuthenticationModuleName` property to `cas`.

   For example:

```
...
  <bean id="authenticationEntryPointFilter"
>
    <property name="authentiationModules">
      <map>
        <entry key="cas" value-ref="casAuthenticationEntryPoint"/>
        <entry key="prestohub"
          value-ref="prestoDefaultAuthenticationEntryPoint"/>
      </map>
    </property>
<property name="defaultAuthenticationModuleName" value="cas"/>
  </bean>
...
```

5. Find the bean with `preauthAuthProvider` ID and:

   a. Comment out the preauthenticated user details property based on `UserDetailsByNameServiceWrapper`.

   b. Uncomment the preauthenticated user details property based on `casAuthenticatedUserDetailsService`.

   For example:

```
  <bean id="preauthAuthProvider"
class="org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvider">
<property name="preAuthenticatedUserDetailsService"
ref="casAuthenticationUserDetailsService"/>
    <!-- property name="preAuthenticatedUserDetailsService">
      <bean id="userDetailsServiceWrapper"
class="org.springframework.security.userdetails.UserDetailsByNameServiceWrapper">
        <property name="userDetailsService" ref="userRepositoryAccessAdapter"/>
      </bean>
    </property -->
  </bean>
```

6. Save your changes to applicationContext-security.xml.

7. Open applicationContext-security-filters-default.xml in any text or XML editor.

   This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.

8. Make sure that the line beginning with `/**/cas/**` is not commented out. Save your changes, if any.

9. Set configuration properties to redirect users to the CAS login form if requests attempt to access Presto directly without a valid CAS ticket. You must:

   a. Open the sso.properties file in any text editor.

This file is located in the *web-apps-home* /presto/WEB-INF/classes folder.

b. Set the following properties for the Presto Server:

- `prestoServerInfo.host` = the host name or IP address for this Presto Server.
- `prestoServerInfo.httpPort` = the HTTP port for this Presto Server. This is 8080 if you installed Presto with default ports.
- `prestoServerInfo.httpsPort` = the HTTPS port for this Presto 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 Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

## Implementing a Custom SSO Filter

If the default extractor and transformer filters available in Presto do not provide the functionality needed to allow Presto to work with your SSO solution, you can create custom filters using the Presto SSO Filter API.

**To use this API**

1. Add the following JARs and classes to your classpath:
   - Classes in the *web-apps-home* /presto/WEB-INF/classes folder.

- The *web-apps-home* /presto/WEB-INF/lib/presto_common.jar file.

2. Implement one or both filters:
   - To create a custom extractor, implement the `SSOTokenExtractor` interface, typically using the `AbstractSSOTokenExtractor` base class.
   - To create a custom transformer, implement the `Transformation` interface.
3. Add these classes to the classpath. Copy either the compiled class file or a JAR containing the compiled class file to one of these folders, respectively:
   - The external configuration folder, if any, for the Presto Server. See "Setting Up an External Presto Configuration Folder" on page 258 for more information.

     **Important:** Deploying additional resources, such as custom SSO filters, to an external configuration folder simplifies future deployments or Presto Server clusters.

   - *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates Presto Server deployments.
   - *web-apps-home* /presto/WEB-INF/lib. This is the default location, but is not recommended as it complicates Presto Server deployments.

## Authentication with Digital Certificates/SSL

There are two aspects of authentication for Presto that you can configure for digital certificates: 1) whether Presto accepts certificates for user authentication and 2) what information Presto uses from the certificates to perform authentication.

Certificate authentication in Presto uses Personal Digital Certificates (PDC) from a client. The default authentication process when Presto 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. Presto 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 Presto Server to use mutual SSL. See "Configure Presto for SSL and Digital Certificates" on page 96 for instructions.
2. Using any text or XML editor, edit the applicationContext-security.xml file in the *web-apps-home* /presto/WEB-INF/classes directory and:
   a. Remove the comment markers from the `<import>` statement for the applicationContext-security-authn-x509.xml file.

      The configuration would look something like this:

```
<beans>
  <import resource="applicationContext-security-authn-rememberme.xml" />
```

```
  <import resource="applicationContext-security-scheduler.xml" />
  <import resource="applicationContext-security-authn-x509.xml" />
  <!--<import resource="applicationContext-security-authn-rsa.xml" /> -->
...
</beans>
```

b. Save your changes to this file.

3. If needed, change the default certificate authentication behavior with one or more of these options:
   - Configure Alternate User ID Extraction to change where Presto obtains the user ID.
   - Configure Dynamic User Support to enable Presto 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 Presto REST API. See "Configure the Presto REST API to Use Certificate Authentication" on page 67 for instructions.
5. If needed, enable certificate caching for the Presto Server.

   By default, the Presto Server does not cache user certificates. This ensures that any changes to user identification or authorization are detected as soon as possible but can impact performance. To turn caching on:

   a. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.

   b. Find the `x509AutheticationProvider` bean.

   c. Add `<property name="certificateCachingEnabled" value="true" />` to the list of properties for this bean.

   d. Save your changes to this file.
6. To apply these changes, restart the Presto Server.

## Configure the Presto REST API to Use Certificate Authentication

By default, certificate authentication is *not* enabled for the REST API to Presto or for Presto Connect for JavaScript (PC4JS).

1. Using any text or XML editor, edit the applicationContext-security-filters-default.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
2. Find the Filter Chain Proxy (`<bean id="filterChainProxy">`) and:

   a. Find the line for `/**/api/rest/**`.

   b. Add `x509ProcessingFilter, `*after*`restLoginProcessingFilter`

   c. In this same bean, find the line for `/**/api*`.

   d. Add `x509ProcessingFilter, `*after*`jumpLoginProcessingFilter`

The result should look something like this:

```
<bean id="filterChainProxy"
>
  <property name="filterInvocationDefinitionSource">
    <value>
      CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
      PATTERN_TYPE_APACHE_ANT
      /**/esd/api/mashsoap/**=basicReqFlowSupportFilter, sessionContextIntegrationFilter, shar
      soapRequestAuthenticationFilter, basicProcessingFilter,  anonymousProcessingFilter, exce
      /**/edge/api/mashsoap/**=basicReqFlowSupportFilter,  sessionContextIntegrationFilter, sh
      soapRequestAuthenticationFilter, basicProcessingFilter,  anonymousProcessingFilter, exce
      /**/api/soap/**=basicReqFlowSupportFilter,  sessionContextIntegrationFilter, sharepointS
      wsSecurityProcessingFilter, basicProcessingFilter, anonymousProcessingFilter, exceptionT
     /**/api/rest/**=restReqFlowSupportFilter,  sessionContextIntegrationFilter, sharepointSSO
      restLoginProcessingFilter, x509ProcessingFiler, basicProcessingFilter,  anonymousProcess
      sessionTimeoutDetectionFilter, exceptionTranslationFilter
     /**/emml/debug=basicReqFlowSupportFilter, sessionContextIntegrationFilter,ssoProcessingFi
     restLoginProcessingFilter, basicProcessingFilter, anonymousProcessingFilter, exceptionTra
     /**/api*= jumpReqFlowSupportFilter, sessionContextIntegrationFilter,ssoProcessingFilter,j
     jumpLoginProcessingFilter, x509ProcessingFiler, basicProcessingFilter, anonymousProcessin
     sessionTimeoutDetectionFilter, exceptionTranslationFilter,filterInvocationInterceptor,ses
      ...
```

This configuration allows both the default HTTP connections with user credentials and HTTPS with digital certificates.

3. Save this change.
4. Open the applicationContext-security.xml file in the *web-apps-home* /presto/WEB-INF/ classes directory.
5. Find the REST Login Processing Filter (`<bean id="restLoginProcessingFilter">`) and add `<property name="ignoreFailure" value="true" />`.

   The bean configuration should look like:

```
<bean id="restLoginProcessingFilter"
>
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="sessionManager" ref="sessionManager" />
  <property name="rememberMeServices" ref="rememberMeServices"/>
  <property name="ignoreFailure" value="true"/>
</bean>
```

6. Find the Authentication Manager (`<bean id="authenticationManager">`) and uncomment the reference to the `x509AuthenticationProvider`.

   The bean configuration should look like:

```
<bean id="authenticationManager"
>
  <property name="providers">
    <list>
      <ref bean="x509AuthenticationProvider"/>
      <ref local="preauthAuthProvider"/>
      <ref local="adminAuthenticationProvider"/>
      <ref bean="defaultAuthenticationProvider"/>
      <ref bean="rememberMeAuthenticationProvider"/>
      <ref local="anonymousAuthenticationProvider"/>
    </list>
```

```
  </property>
</bean>
```

7. Save your changes to this file.

## Configure Alternate User ID Extraction

You can use regular expressions to define other portions of the certificate's subjectDN as the source for the user's ID. To define an alternate location for the user ID:

1. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
2. Find the x509 Authorities Populator (`<bean id="x509AuthoritiesPopulator" >`) and:
   a. Remove the comment markers around the `<property name="subjectDNregex">` element.
   b. Change the regular expression in the `<value>` to define what to use as the user ID.
3. Save your changes to this file.

## Configure Dynamic User Support

Dynamic user support allows Presto to accept a valid certificate even if the user is not provisioned in the User Repository. You can also define a set of roles for dynamic users to enable specific service access. To configure dynamic user support:

1. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
2. Find the x509 Authorities Populator (`<bean id="x509AuthoritiesPopulator" >`) and:
   a. Change the value attribute for `<property name="dynamicUser"/>` to `IN_MEMORY`.

      For example:

      ```
      <property name="dynamicUser" value="IN_MEMORY"/>
      ```

   b. To set up groups or roles to use as authorization for dynamic users, add or uncomment `<property name="dynamicUserRoles">`.
   c. Add a `<list>` child and add a `<value>` child under that for each group or role that dynamic users should be authorized for.

      For example:

      ```
      <property name="dynamicUserRoles">
        <list>
          <value>EditPreferences</value>
          <value>ViewNews</value>
        </list>
      </property>
      ```

3. Save your changes to this file.

## Configure Additional Certificate Validation

You can have certificate authentication perform additional validation beyond simple user ID checks.

1. Implement the additional validation logic in a class that implements the com.jackbe.jbp.sas.security.x509.x509CertValidator interface.

   To do this, add the following JARs and classes to your classpath:

   - Classes in the *web-apps-home* /presto/WEB-INF/classes folder.
   - The *web-apps-home* /presto/WEB-INF/lib/presto_common.jar file.

   See the *Custom Certificate Validation API* for details on implementing this interface.

   Then add your custom class to the classpath in one of these folder:

   - The external configuration folder, if any, for the Presto Server. See "Setting Up an External Presto Configuration Folder" on page 258 for more information.

     **Important:** Deploying additional resources, such as custom validation classes, to an external configuration folder simplifies future deployments or Presto Server clusters.

   - *web-apps-home* /presto/WEB-INF/classes. This is the default location, but is not recommended as it complicates Presto Server deployments.
   - *web-apps-home* /presto/WEB-INF/lib. TThis is the default location, but is not recommended as it complicates Presto Server deployments.
2. Using any text or XML editor, edit the applicationContext-security-authn-x509.xml file in the *web-apps-home* /presto/WEB-INF/classes directory.
3. Find the x509 Authentication Provider (`<bean id="x509AuthenticationProvider" >`) and:
   a. Find the <property name="validators"> element.
   b. Add a <list> child and add a <bean> child with your implementation class name.

      For example:

```
<bean id="x509AuthenticationProvider">
    ...
    <property name="validators">
      <list>
        <bean/>
      </list>
    </property>
    ....
</bean>
```

4. Save your changes to this file.

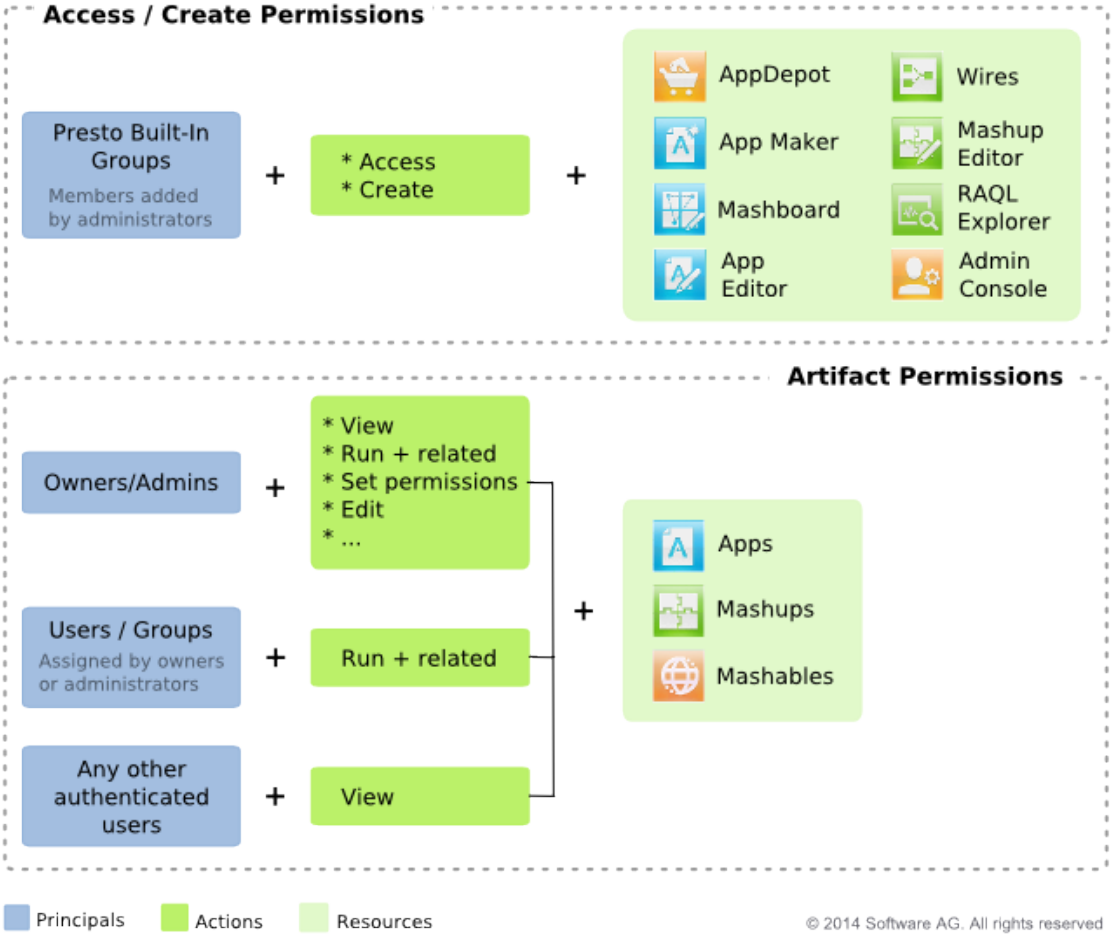
# Authorization Policies and Permissions

Authorization policies determine the actions that users can perform with the mashables, mashups and apps that Presto governs. Policies also determine user access to the features and tools in the Presto Hub and the Presto Enterprise AppDepot.

By default, authorization is enabled in Presto. 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 77 for instructions.

The categories of authorization policies that are defined in Presto are shown below.




- **Access/Create Permissions**: are defined using Presto built-in user groups as the principals. See the "Built-In Presto User Groups and Permissions" on page 73 topic for detailed information these policies.

To grant access to Presto tools and enable users to create artifacts in Presto Hub, you add users to these built-in groups. See "Grant User Access to Presto with Built-in Groups" on page 72 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 Presto 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 75. 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 56 for instructions.

  Users also get several other related permissions when you grant run permissions. See the "Built-In Presto User Groups and Permissions" on page 73 topic for more information on the additional permissions granted with run.

- **View Permissions**: authenticated users can see artifacts in Presto 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 76 for information.

## Grant User Access to Presto with Built-in Groups

All users in the Presto User Repository automatically belong to the `Presto_AuthenticatedUsers` built-in group which has permission to access the PrestoAppDepot and work with any apps to which they also have been granted run permissions. To enable users to work in Presto 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 Presto User Groups and Permissions" on page 73 topic for information on the specific access policies for these groups. Or use the "Default User Accounts" on page 58 in Presto to better understand the permissions for these groups.

- If you are using the Default User Repository with Presto, both groups and users are defined with the Admin Console. To grant users permissions with the Presto built-in groups:

1. Add users to the Presto Repository. See "Create Users" on page 47 for instructions.
2. Assign users to the appropriate built-in groups. See "Edit, Grant Permissions and other User Management Tasks" on page 48 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 50 for instructions.

- If you have configured Presto to use your LDAP Directory as the User Repository, you relate users to the Presto built-in groups in LDAP. To grant users permissions with the Presto 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 Presto built-in permissions, you add these predefined names to your LDAP Directory. Mapping from configuration in Presto 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 JackBe sales representative.

  2. Assign users to these new groups in LDAP.

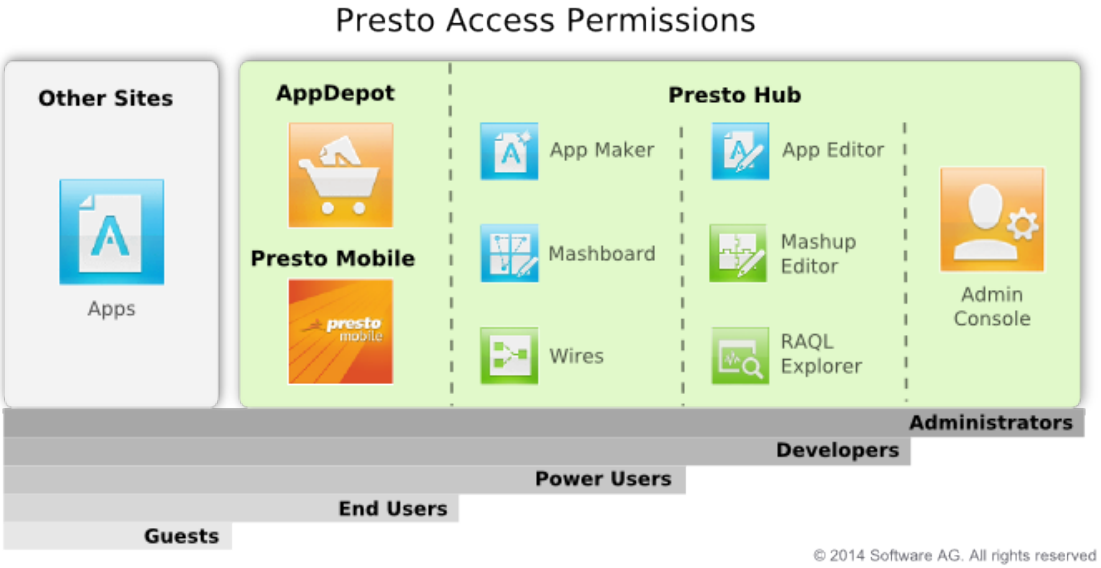
# Built-In Presto User Groups and Permissions

Presto has a set of built-in user groups that define access permissions to the various features in Presto 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 Presto Built-In Groups" on page 73 and "Artifact Permissions for Users with Run Permissions" on page 75.

## Access Policies Using Presto Built-In Groups

A high-level view of access policies for the built-in groups is shown below:




- *Guests* = users in other sites who are not authenticated. Guests can work with apps deployed in other sites if the app and all other artifacts that it depends on have granted run permissions to the `Presto_Guest` built-in group. See "Enabling Guest Access" on page 58 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 Presto user in the AppDepot and in Presto Hub.

- *End Users* = all authenticated users (in the Presto Repository) that are not in another built-in group. Authenticated users can access Presto 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 Presto 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 Presto 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 Presto extension points. Developers also have access to other technical information, such as the Technical Specification for mashables and mashups or the API Console.

This group is typically used for IT or line-of-business developers involved in developing apps, mashups or mashables for specific projects. Developers may also develop other extension features to provide specific capabilities in Presto Hub for power users.

- **Administrators** = users in the `Presto_Administrator` group have unrestricted permissions in Presto. They can work with any tools, features or artifacts. They also have permissions to use the Admin Console to configure and manage Presto 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 Presto 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

Presto 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 Presto 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 `Presto` as the search term and search for groups to get a list of the built-in Presto 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 Presto groups that you assign to users. View permissions determine what artifacts appear in search results in Presto Hub and the AppDepot and the activity feed in the Presto 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 Presto Hub can also see any mashable, mashup or app in search results in Presto 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, Presto displays an error message. Users must request run permissions for the artifact from a Presto 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 Presto 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 Presto. You can disable authorization checks to simplify workflow during development.

**To enable/disable authorization**

1. Click  Admin Console in the Presto 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**.

# 4 Presto Server Configuration

### Basic Configuration and Logging

- "Manage Licenses for Presto, Universal Messaging and BigMemory" on page 23
- "Support International Character Sets and Locales" on page 85
- "Update Help or Use Offline Help for Presto" on page 88
- "Change the Presto HubTheme" on page 89
- "Configure the Presto Server with Custom Ports" on page 92
- "Configure the Presto Server to Work with a Proxy Server" on page 93
- "Define a Proxy Server Whitelist for Presto" on page 94
- "Configure Presto for SSL and Digital Certificates" on page 96
- Configuring "Presto Notifications" on page 106
- Configuring "Presto Logging" on page 103 and viewing logs
- "Manage Data Sources and Drivers" on page 131
- "Manage Categories for Mashups, Mashables and Apps" on page 136
- "Manage Providers for Mashups, Mashables and Apps" on page 137
- "Work With Presto Attributes" on page 137

See also "Presto and MashZone Repositories" on page 197 for links to configuration for the Presto Repository.

### Mashables, Mashups and Wires Configuration

- "Manage Artifact Attributes" on page 140
- "Manage Data Sources and Drivers" on page 131
- "Configure the Default Operations Generated for Database Mashable" on page 134
- "Disable Mashup Features" on page 141
- "Configure Mashable HTTP Request Timeouts" on page 143
- "Configure HTTP Response Header Forwarding" on page 142
- "Enable or Disable the Snapshot Feature" on page 143
- "Set Web Feed Normalization" on page 144
- "Handle SOAP Encoding Errors for WSDL Services" on page 144
- "Add XML Schemas to the Wires Mapper Block" on page 145

### Add-Ons, Performance, Deployment and Miscellaneous

- "BigMemory for Caching, Connections and Presto Analytics" on page 114

- "Configure Presto Connections to SharePoint" on page 109
- "Memory Configuration for the Presto Server" on page 81
- "Upgrade to New Versions of Presto and Migrate Artifacts" on page 211
- "Deploying Presto Instances, Clusters or Artifacts" on page 230
- See also "Presto Security" on page 53 for links to server configuration tasks for authorization.

# Memory Configuration for the Presto Server

Presto 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, Presto is no longer supported on 32-bit architectures which have memory access limitations from Java.

If you are working with large datasets in Presto Analytics or you are deploying Presto 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:

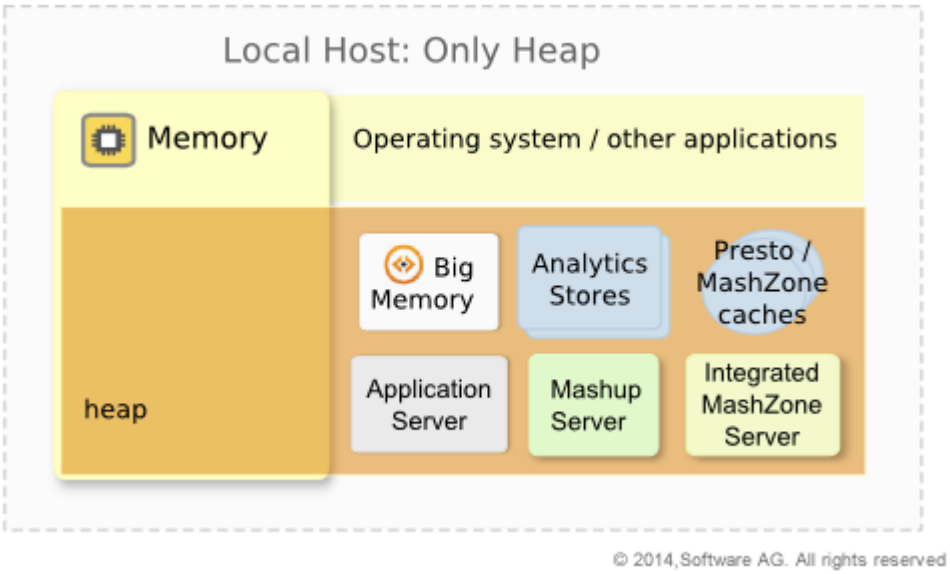
| If | See |
|---|---|
| ■ Your computer has less than 4G of RAM memory, or<br>■ You have *not* installed BigMemory Server(s). | "Configuration When Presto Uses Only Heap Memory" on page 81 |
| ■ You have installed BigMemory Server(s), or<br>■ Your computer has more than 4G of RAM memory. | "Configuration When Presto Uses Heap and Off-Heap Memory" on page 83 |

## Configuration When Presto Uses Only Heap Memory

The initial default Java heap memory settings for Presto are appropriate for small applications or development environments 1G as the maximum heap size. Both the Integrated MashZone Server and the Event Service that are deployed with Presto also use this heap space.

**Note:** If you are using Presto Analytics with large datasets and limited memory (less than the recommended 4G minimum), configuring BigMemory to use off-heap memory is *not* recommended as it can adversely affect performance.

Some portion of available memory must be reserved for the operating system and any other applications on this host, as shown in the following figure:




How much memory to allocate to Presto depends on available memory, requirements for the Integrated MashZone Server and Event Service and what other applications may run on this computer.

**Note:** For more information and suggestions on memory requirements for the Integrated MashZone Server and Event Service, see "Tune Memory/Caching for the Integrated MashZone Server" on page 149.

**To update memory configuration**

1. In a text editor of your choice, open the application server configuration file appropriate for your operating system:
   - *presto-install* /apache-tomee-jaxrs/bin/setenv.bat, for Windows
   - *presto-install* /apache-tomee-jaxrs/bin/setenv.sh, for Linux, OS/X or UNIX
2. Change any of these Java memory options:

| | |
|---|---|
| -Xms | Default = `512M` |
| -Xmx | Default = `1G` |

| | |
|---|---|
| -XX:PermSize | Default = `128M` |
| -XX:MaxPermSize | Default = `256M` |

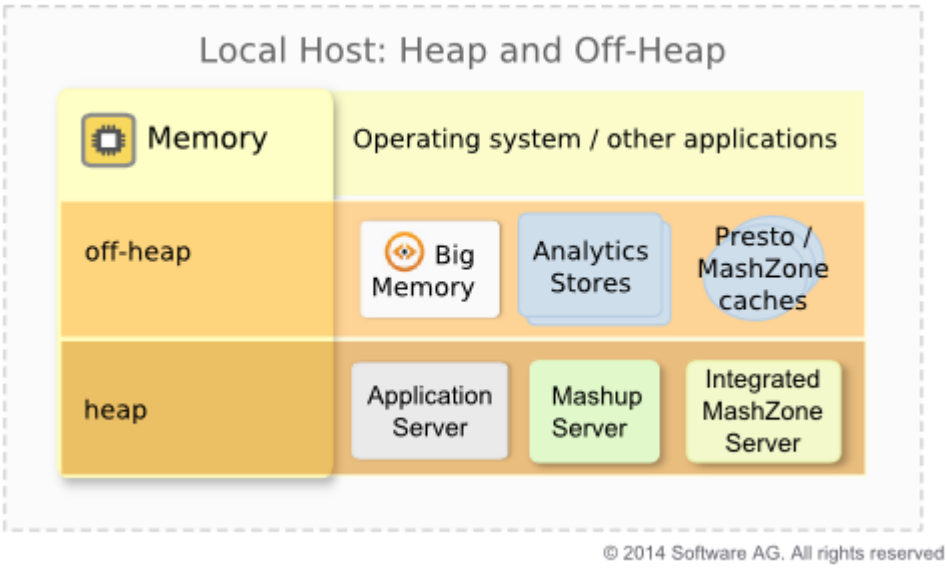
3. Save your changes and restart the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

## Configuration When Presto Uses Heap and Off-Heap Memory

Presto 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 Presto to use off-heap memory. See "Manage Licenses for Presto, Universal Messaging and BigMemory" on page 23 for instructions.

With combination heap and off-heap memory, as this figure shows, BigMemory uses off-heap memory for the Presto Analytics In-Memory Stores and Presto caches. All other Presto processing, including the Integrated MashZone Server and Event Service that are deployed with Presto, remains in heap.




The total available off-heap memory may be limited to local off-heap memory as shown above, or it may include additional off-heap memory on external hosts if you have installed BigMemory Server arrays.

**Note:** For more information and suggestions on memory requirements for the Integrated MashZone Server and Event Service, see "Tune Memory/Caching for the Integrated MashZone Server" on page 149.

To update memory configuration:

1. In a text editor of your choice, open the application server configuration file appropriate for your operating system:
   - *presto-install* /apache-tomee-jaxrs/bin/setenv.bat, for Windows
   - *presto-install* /apache-tomee-jaxrs/bin/setenv.sh, for Linux, OS/X or UNIX
2. Change or add either of these memory options used with BigMemory:

| | |
|---|---|
| -Dpresto.bm.maxOffHeap | Default = `1G`<br>This is the maximum size of *local* off-heap memory that BigMemory can use for the Presto Analytics In-Memory Stores and Presto caches.<br>This property sets off-heap memory limits in the *presto-config* /ehcache.xml configuration file. The total size of off-heap memory may include additional, external memory depending on how BigMemory is deployed. |
| -XX:MaxDirectMemorySize | Default = `1500M`<br>This Java memory option must be set to allow access to both off-heap and an additional allocation for Java.<br>The value of this option must *always* be larger than the memory allocated to off-heap. A good rule of thumb is at least 500M more. |

3. Change or set any of these Java memory options:

| | |
|---|---|
| -Xms | Default = `512M` |
| -Xmx | Default = `1G` See the Java Tuning White Paper for more information and suggestions. |
| -XX:PermSize | Default = `128M` |
| -XX:MaxPermSize | Default = `256M` |

4. Save your changes and restart the Presto Server. See "Start and Stop the Presto Server" on page 20 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, Presto must know the character set and how it is digitally represented - the *character encoding*.

**Note:** Presto 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 Presto properly handles and presents data to users. The areas of configuration involved in this support include:

- *Presto Repository*: the character encoding for this repository determines what character sets users can use when they create artifacts. The timezone for the Presto Repository also affects timestamps shown in Presto Hub.

  For configuration options and instructions, see "Presto Repository Encoding and Timezone" on page 86.

- *Artifact data*: Presto 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 87.

- *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 87 for details.

- *Logging*: you can also support international characters and different locales for the messages sent to Presto logs. See "Message Log and Default Locales" on page 87 for details.

# Presto Repository Encoding and Timezone

### Set the Repository Character Encoding

The character encoding for the Presto 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 Presto and MashZone Repositories to Microsoft SQL Server
  - Move the Presto and MashZone Repositories to MySQL
  - Move the Presto and MashZone Repositories to Oracle
  - Move the Presto and MashZone Repositories to PostGres

### Set the Repository Timezone or Offset

The default timezone that is used to record timestamps such as the created date and time for an artifact is the timezone for the host of the Presto 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 Presto Repository uses to match the timezone for the Presto Server. See "Synchronize the Presto Repository and Presto Server Time Zones" on page 200 for instructions.
- Configure the display timezone in Presto Hub as an offset from UTC. Note that this does not affect the actual timezone recorded in the Presto Repository.

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

  Edit this JSON property, setting the number of minutes as a UTC offset. For example, `300` sets this to Eastern Standard Time while `-180` sets this to Arabic Standard Time.

  Once the property is saved, restart the Presto 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. Presto uses UTF-8 as the default character encoding for mashable and mashup results.

For mashables that are file-based, Presto 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 (*presto-install* /apache-tomee-jaxrs/bin/setup.[bat|sh]).

You can provide a mechanism to override this default encoding for individual mashables by defining an artifact attribute named `encoding_override`.

### 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 Presto. If the user's locale is not available or it is not a supported locale, Presto uses the locale from the JVM for the Presto 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 Presto. See "Message Log and Default Locales" on page 87 for instructions.

## Date, Time and Numeric Display Options

In general, Presto 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, Presto use the default system locale which typically is EN_us.

Some built-in views in Presto, 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

Presto uses the default locale defined for the JVM for all messages that are added to Presto 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 Presto using Java properties**

1. In a text editor of your choice, open the setenv.bat file, for Windows systems, or the setenv.sh file, for Linux, OS/X or UNIX systems, in the *presto-install* /apache-tomee-jaxrs/bin folder.
2. Add a line just below the initial lines which set the `JAVA_OPTS` environmental variable.
   - For Windows, add:

     ```
     set JAVA_OPTS=%JAVA_OPTS% -Duser.county=country-code-
     Duser.language=language-code
     ```

   - For Linux, OS/X or UNIX, add:

     ```
     JAVA_OPTS="$JAVA_OPTS -Duser.county=country-code-
     Duser.language=language-code"
     ```

   For example:

   ```
   JAVA_OPTS="-XX:PermSize=128M -XX:MaxPermSize=256M  -Xms512m  -Xmx1G"
   JAVA_OPTS="$JAVA_OPTS -DPRESTO_HOME=$CATALINA_BASE"
   JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8"
   JAVA_OPTS="$JAVA_OPTS -Dpresto.raql.udf.libsReload=false
     -Dpresto.raql.udf.libsDir=$CATALINA_BASE/../raql-udfs"
   JAVA_OPTS="$JAVA_OPTS
     -Dlog4j.configuration=file://$CATALINA_BASE/conf/log4j.properties"
   JAVA_OPTS="$JAVA_OPTS -Duser.country=CA -Duser.language=fr"
   # (this is set by the installer)
   JAVA_HOME=${JAVA_HOME}
   ...
   ```

3. Save your changes to the file.
4. Restart the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

# Update Help or Use Offline Help for Presto

By default, help buttons within Presto point to online topics in the Presto Library based on a configuration property defined in the Presto Repository. You must update this configuration property if you migrate to a new Presto release but continue to use your existing Presto Repository.

You must also update this property if you want help in Presto to point to an offline copy of the Presto Library. This is useful if your organization has network restrictions that prohibit access to external networks.

- To update help configuration to the online Presto Library for a new Presto release:
  1. Find the exact URL to use in the Migration section of the Presto Release Notes for the new release.
  2. Click  Admin Console in the Presto Hub main menu.

3. Open the Server section and click **Help Library**.
4. Enter the base URL found in the first step.
5. Save your changes.

- To update configuration for offline access:

  1.
  2. Unzip the Presto Library and host it in a web server or application server that the Presto Server can access.
  3. Click  Admin Console in the Presto Hub main menu.
  4. Open the Server section and click **Help Library**.
  5. Enter the base URL to the offline Presto Library *including a trailing* /.

     If you unzipped the offline Presto Library in a web server with an address of `http://192.168.1.5:80/`, for example, the URL to enter as the base URL would be `http://192.168.1.5:80/prestodocs-offline/`.
  6. Save your changes.

## Change the Presto HubTheme

You can change the following aspects of the main Presto 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.

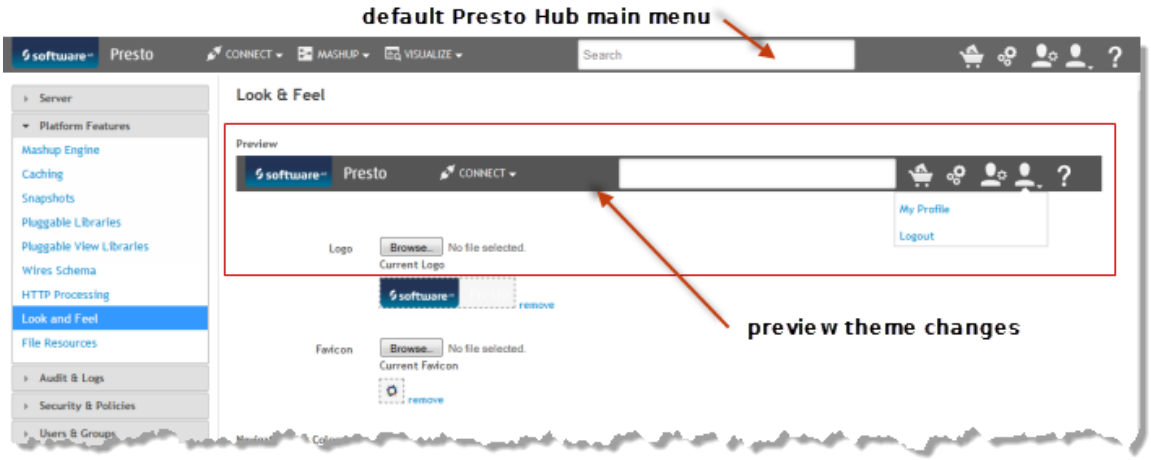
- The color to use for the text of main menu items.

  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.s

**To change the main menu theme**

1. Click  Admin Console in the Presto 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 Presto Hub menu.

   To remove a custom logo and return to the default Presto logo, click **Remove** next to the current logo image in the page. The browser automatically refreshes.
4. To change the background color for the menu and use a flat look:
   a. Use the color picker for the Top Gradient color and select the color you want.
   b. Choose the same color for the Bottom Gradient color.

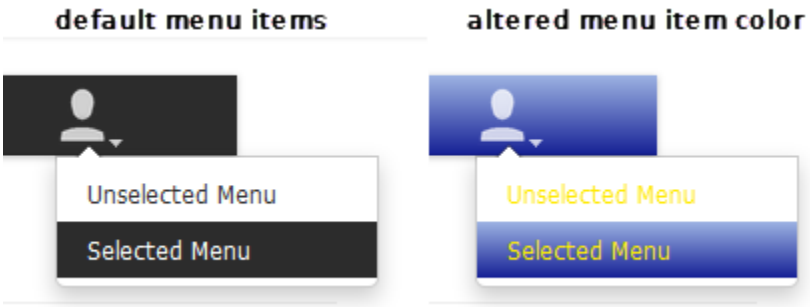
   The menu preview updates showing you what this will look like:


5. To change the background color for the menu and use a curved look, use a gradient:
   a. Use the color picker for the Top Gradient color and select the color you want. Typically this is a lighter color than the bottom gradient.
   b. Choose a different color for the Bottom Gradient color. Typically this is a darker color than the top gradient.

The menu preview updates showing you what this will look like:


6. To change the color of menu items, use the color picker for **Menu font color**.

   This single color is used for both selected and unselected item so it is better to choose a color that is easily visible against both white and the menu background color(s).




7. Click **Save Changes** to apply the new theme.

## 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 Presto 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.

# Configure the Presto Server with Custom Ports

Port configuration is initially set when you install Presto. If you change these ports or you need to host multiplePresto Servers on one host, you must update configuration in thePresto Server. You may also need to change ports for the Presto Repository and for TomEE.

## Change Presto Server Ports

The host name and port for thePresto Server defaults to localhost and 8080 respectively. The port is typically defined in configuration for TomEE, the application server that hosts Presto.

To change the host and port information, you must:

1. Update port configuration for the application server that hostsPresto in *presto-install* /apache-tomee-jaxrs/conf/server.xml in the `<Connector>` elements for HTTP and/or HTTPS.
2. Update host and port information used for user email notifications.

   **Note:** This configuration is used to generate full, correct links in notifications that are sent to users via email. It does not affect the Presto 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 Presto Server.See "Start and Stop the Presto Server" on page 20 for instructions.

## Change Presto Repository Ports

If you are running multiple instances of the Presto Server in one host with separate instances of the Presto Repository, you may need to use different ports for each database instance:

| | |
|---|---|
| *The default Presto Repository* | No updates to ports are needed as the Derby database is embedded. |

| | |
|---|---|
| *The Presto Repositoryis hosted in a robust database* | You must use different ports for each Presto Repository instance. |

To update Presto Repository ports in the Admin Console.

1. Click Admin Console in the Presto Hub main menu.
2. Expand the **Presto 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 Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

## TomEE Application Server Port

If you are running multiple instances of the Presto Server in one host, you must have separate application server instances for each. You must update the following ports:

| Configuration File | Element |
|---|---|
| *presto-install* /apache-tomee-jaxrs/conf/server.xml | `<Server>` to set the command port<br>`<Connector> for the HTTP port`<br>`<Connector> for the HTTPS port` |

# Configure the Presto Server to Work with a Proxy Server

Presto is *only* compatible with HTTP proxy servers.

If you have a proxy server in your environment that Presto should use, you *must* add configuration information to the Presto 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 Presto" on page 94 for more information.

### To configure a proxy server

1. Start the Presto Server.

   See "Start the Presto Server" on page 21 for instructions.
2. Open Presto 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 Presto 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 Presto Server should use to connect to the proxy server. This is only required if your proxy server requires credentials.
   - *Password* = the password that the Presto 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 Presto" on page 94 for instructions.
9. Click **Save proxy settings**.

## Define a Proxy Server Whitelist for Presto

If you have configured a proxy server for the Presto 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 Presto 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 Presto Server should access without the proxy server. To use wildcards in IP addresses, see "Using Regular Expressions in a Whitelist" on page 95.
   - *Bypass Host List* = enter one or more fully-qualified host names, separated by commas, that the Presto Server should access without the proxy server.To use wildcards in IP addresses, see "Using Regular Expressions in a Whitelist" on page 95.

- *Bypass Domain List* = enter one or more domain names, separated by commas, that the Presto Server should access without the proxy server. To use wildcards in IP addresses, see "Using Regular Expressions in a Whitelist" on page 95.

4. Click **Save proxy settings**.
5. Log out of the Presto Hub.
6. Stop and restart the Presto 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
```

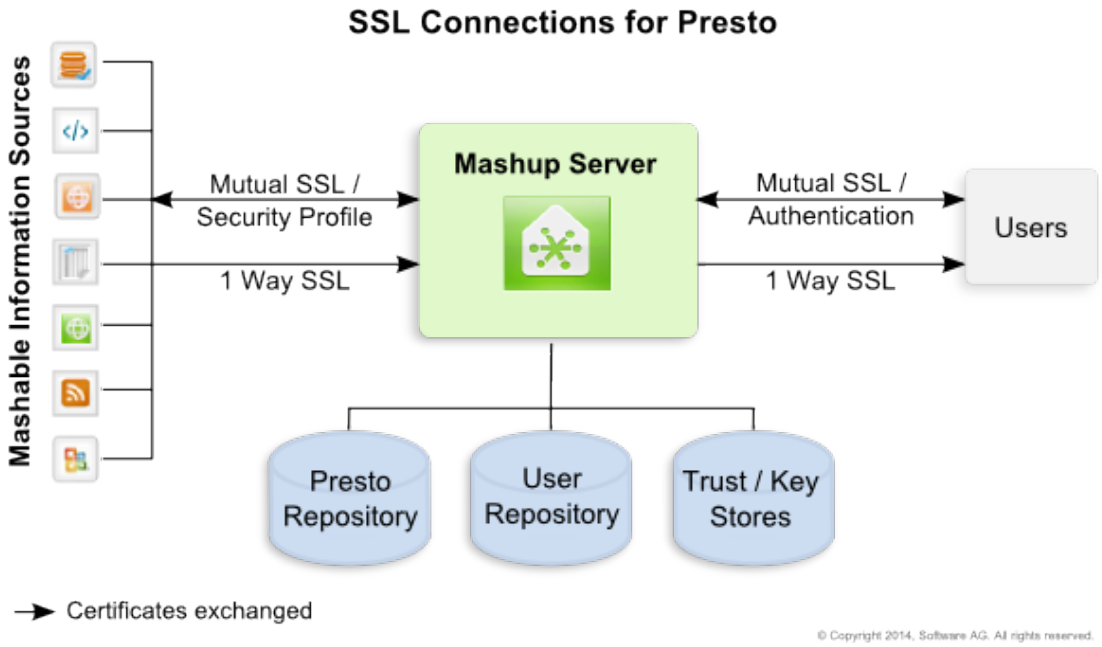
Note that an expression of `.*.customers.mydomain.com` would also match these same three hosts.

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 95 for more information.

## Configure Presto for SSL and Digital Certificates

Presto expects HTTP as the default transport protocol from clients to the Presto Server. Connections from the Presto Server to mashable information sources typically also use HTTP.

Presto 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. Presto can also use digital certificates from clients in user authentication.




The certificate store, certificates and configuration needed to support SSL in Presto depends on the connection requirements, as shown below:

| | Certificate Store and Certificates | | Store Configuration | | | Presto Configuration | |
|---|---|---|---|---|---|---|---|
| | Key | Trust | Java | App Server | Presto | Authentication | Security Profiles |
| "One-Way SSL to Presto" on page 100. | ✓ | | | ✓ | | | |
| Mutual to Presto<br><br>See "Configure Mutual SSL Between Users and Presto" on page 99. | ✓ | ✓ | | ✓ | | ✓ | |

| | Certificate Store and Certificates | | Store Configuration | | | Presto Configuration | |
|---|---|---|---|---|---|---|---|
| | Key | Trust | Java | App Server | Presto | Authentication | Security Profiles |
| "One-Way SSL to Mashable Information Sources" on page 101. | | ✓ | can be in either | | | | |
| "One-Way SSL to Information Sources Using <directinvoke> in Mashups" on page 101. | | ✓ | ✓ | | | | |
| Mutual to Mashables<br>See "Configure Mutual SSL Between Presto and Mashable Information Sources" on page 100 | ✓ | ✓ | | | ✓ | | ✓ |

See also "The Certificate Store and Certificates" on page 98 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 Presto, the key certificate pair stored in the key store identifies the Presto Server to users, for both one-way and mutual SSL. The key certificate pair identifies the Presto Server to mashable information sources for mutual SSL.

You must generate a key certificate pair for Presto. 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 Presto or you can use separate certificate stores. You can use an existing certificate store for Presto, 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 Presto

The Presto Server and users both exchange certificates. Presto 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 Presto Server.
  - A key certificate pair for the Presto 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 99 for more information.

See "The Certificate Store and Certificates" on page 98 for more information.

- Configuration in the application server hosting Presto to use the HTTPS port. This also includes configuration identifying the key store and trust store for the Presto Server. See "Configure HTTPS and Certficate Stores in the Application Server" on page 101 for instructions.
- Optional configuration in Presto to use digital certificates for user authentication. See "Authentication with Digital Certificates/SSL" on page 66 for instructions.

## Configure Mutual SSL Between Presto and Mashable Information Sources

Both the information source and Presto 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 Presto. It requires:

- Store and Certificates:
  - A certificate store as key store and trust store for the Presto Server.
  - A key certificate pair for the Presto 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 99 for more information.

  See "The Certificate Store and Certificates" on page 98 for instructions.
- Configuration in Presto for both the key store and trust store. See "Configure Certificate Stores in Presto" on page 102 for instructions.
- Security Profile configuration for each mashable information source. You provide this configuration when you register the mashable.

## One-Way SSL to Presto

This requires:

- A key store and a key certificate pair for Presto. See "The Certificate Store and Certificates" on page 98 for more information.

- Configuration in your application server for the HTTPS port to Presto and the key store. See "Configure HTTPS and Certficate Stores in the Application Server" on page 101 for instructions.

## One-Way SSL to Mashable Information Sources

This requires:

- A trust store for Presto. See "The Certificate Store and Certificates" on page 98 for more information.
- Configuration for the trust store in either:
  - The application server hosting the Presto Server. See "Configure HTTPS and Certficate Stores in the Application Server" on page 101 for instructions.
  - Java. See "Update SSL Configuration for Java" on page 103 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 99 for more information.

## One-Way SSL to Information Sources Using <directinvoke> in Mashups

This requires:

- A trust store for Presto. See "The Certificate Store and Certificates" on page 98 for more information.
- Configuration for the trust store in Java. See "Update SSL Configuration for Java" on page 103 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 99 for more information.

## Configure HTTPS and Certficate Stores in the Application Server

Configuration for SSL for Presto can be defined in the application server that hosts the Presto 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 Presto Server, find or create these stores and certificate. See "The Certificate Store and Certificates" on page 98 for instructions.
2. Configure Tomcat for secure connections from clients to the Presto 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 Presto Server to listen to that port. See "Configure the Presto Server with Custom Ports" on page 92 for more information on this step.
3. If needed, enable Presto authentication to use certificates. See "Authentication with Digital Certificates/SSL" on page 66 for instructions.

## Configure Certificate Stores in Presto

For secure connections to mashable information sources using mutual SSL and SSL security profiles, you must add key store and trust store configuration to Presto in the Admin Console:

1. Click  Admin Console in the Presto 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 Presto Server or other Presto 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 Presto 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 Presto Server:

1. Open the appropriate startup script (below) for the application server hosting the Presto Server in any text editor:
   - *presto-install* /apache-tomee-jaxrs/bin/setenv.bat file for Windows environments
   - *presto-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX environments
2. Add these system properties in the Java options in this script:
   - `-Djavax.net.ssl.trustStore=`*`/path/to/mashup-server/truststore`*: this is the absolute path to the trust store for the Presto Server.
   - `-Djavax.net.ssl.trustStorePassword=`*`truststore-password`*: this is only required if the Presto Server's trust store uses a password.
3. Save your changes to the script and restart the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

# Presto Logging

In addition to logging from your application server, Presto provides the following types of logging:

- Basic logging for Presto Server startup, shutdown and exceptions based on a configured logging level. See "Configure Logging for the Presto Server" on page 104 and "View the Presto Server Log" on page 204 for more information.

- Audit logging tracks the invocation of each mashable or mashup. See "Turn Audit Logging On or Off" on page 105, "Purge the Audit Log" on page 206 and "View the Audit Log for a Mashable, Mashup or App" on page 205 for more information.

## Configure Logging for the Presto Server

The Presto Server log, prestoserver.log, logs all exceptions from startup through shutdown. See "Presto Logging" on page 103 for links to additional types of logging you can use with Presto.

**Note:** For clustered environments, updating logging configuration affects logging only for the Presto Server where you are currently logged in. Generally, this is the behavior you want.

To change logging for all Presto Servers in the cluster, update logging configuration for one server and copy the updated *presto-install* /apache-tomee-jaxrs/conf/log4j.properties file to each of the other Presto Servers in the cluster. You do not need to restart Presto Servers.

**To configure basic logging for the server**

1. Click  Admin Console in the Presto 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 Presto 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 Presto technical support to help debug specific issues. It is the logging level to use in the data mapping layer for Presto.
   - *HTTP client logging level* = This property should only be changed when requested by Presto technical support to help debug specific issues. It is the logging level to use for requests/responses between the Presto Server and mashable information sources.

- *NET SF logging level* = This property should only be changed when requested by Presto 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 Presto technical support to help debug specific issues. It is the logging level to use with the Presto security layer.
- *Apache logging level* = This property should only be changed when requested by Presto technical support to help debug specific issues. It is the logging level to use with many of the third party libraries used in Presto.
- *Spring framework logging level* = This property should only be changed when requested by Presto 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 Presto 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 Presto. For more information, see http://logging.apache.org/log4j/1.2
5. Click **Save log settings**.

   This change becomes effective automatically within 60 seconds.

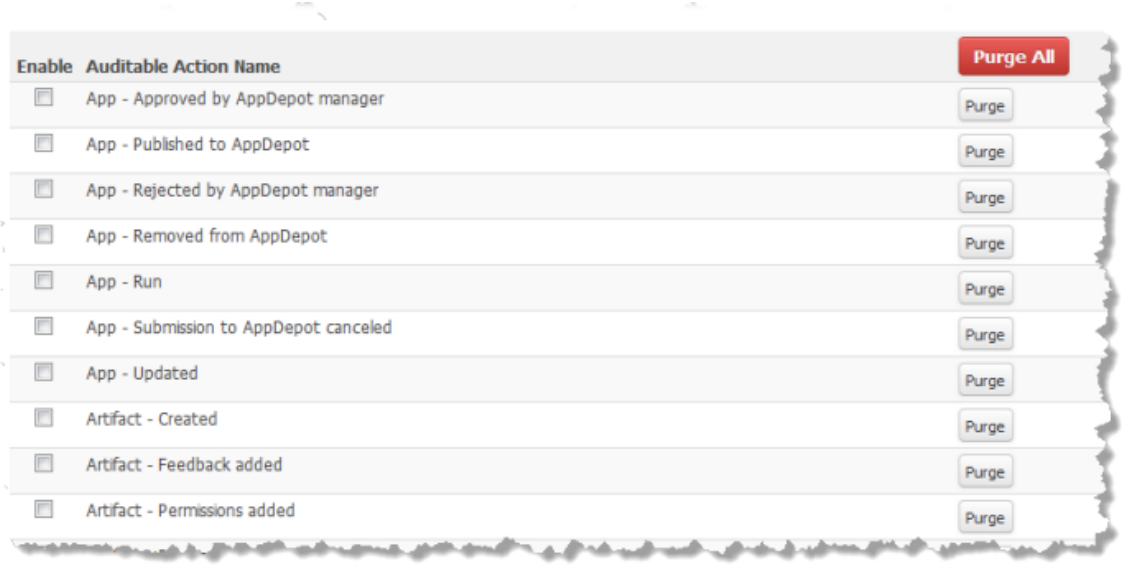
## Turn Audit Logging On or Off

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 Presto 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.

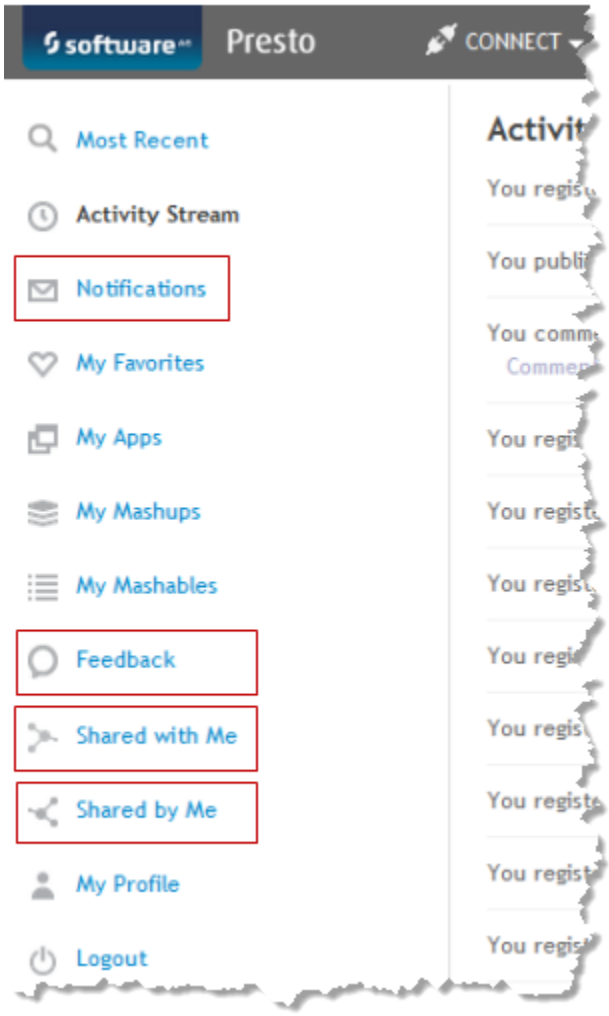
3. To turn audit logging on for a specific event, set the **Enable** option for that event.

> **Note:** Audit events are tracked in a table in the Presto 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 206.

4. To turn audit logging off for a specific event, clear this option for that event.

## Presto Notifications

Presto 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 Presto Hub from the home page:




You can also configure Presto to send notifications as email messages.

**To support email notifications**

1. Configure an email server in Presto. See "Configuring a Mail Server for Presto" on page 108 for instructions.
2. Optionally define the host and port to use for Presto in links within email notices. See "Configure the Presto Server with Custom Ports" on page 92 for more information.
3. Turn on email notifications in the Admin Console:
   a. Click Admin Console in the Presto 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 Presto User Repository, Presto 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 Presto to look for the correct LDAP user attribute. See "Update the User Email Attribute from LDAP" on page 108 for instructions.

## Configuring a Mail Server for Presto

By default, Presto sends notices for a variety of administrator and user actions within Presto 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 Presto Server for your mail server.

1. Click  Admin Consolein thePresto Hubmain menu.
2. In the Server section, select **Mail Server**.
3. Complete the connection properties:
   - **Host** = your mail server domain name or host IP address.
   - **Port** = the port number for your mail server. This defaults to 25, for SMTP.
4. Set the **Requires authentication**if your SMTP server requires credentials and complete these properties:
   - **Username** = the user account the Presto Server should use to connect to your mail server.
   - **Password** = the password the Presto 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 Presto is configured to use your LDAP Directory as the User Repository, Presto 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 Presto to ensure that email notifications are successfully delivered.

**To update the mail attribute**

1. Click  Admin Console in the Presto Hub main menu.

2. Expand the **Presto Repositories** section and click **User Repository - LDAP**.
3. Click **Advanced Options** and scroll down to find Presto 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 Presto Attributes** and **Attributes Used in Wildcard Search** fields as these properties commonly include the `mail` attribute.
6. Click **Save the changes**.
7. Restart the Presto Server to apply these updates.

# Configure Presto Connections to SharePoint

If you have the Presto Add-On for SharePoint, you must configure connections in Presto from the Presto Server to SharePoint. You must also configure connections in SharePoint from SharePoint to the Presto Server.

This configuration determines:

- How many Presto Add-On for SharePoint licenses are needed.

  The Presto 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 Presto Servers and SharePoint" on page 109 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 Presto 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 Presto to be registered as mashable information sources or used as information sources in mashups.

  This is determined by the connections you define in Presto from the Presto Server to SharePoint farms and site collections. Users can also register SharePoint lists in SharePoint as mashables in Presto, so this only affects users in Presto.

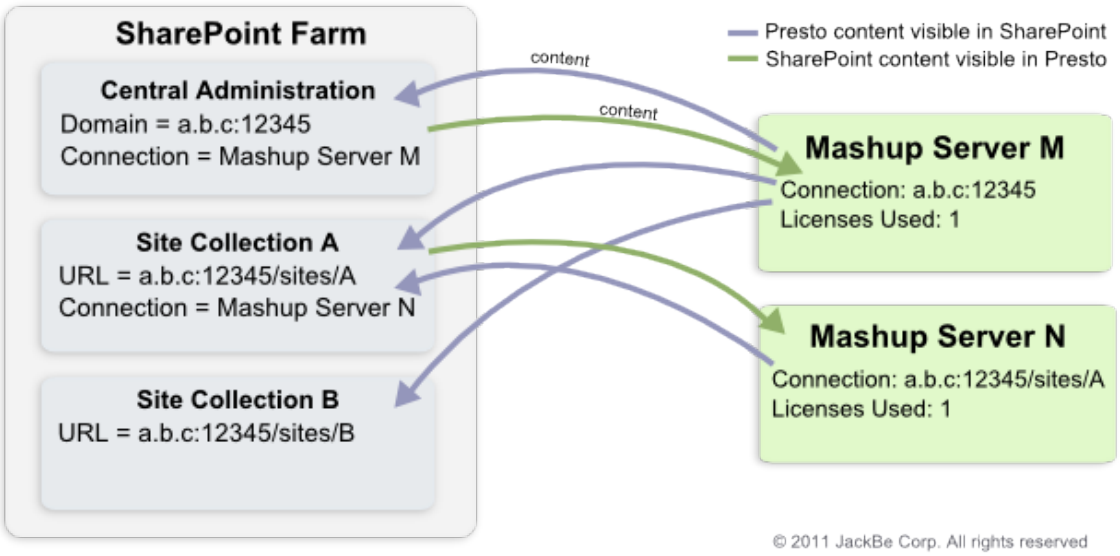
## Connection Patterns Between Presto Servers and SharePoint

How you define connections between Presto and SharePoint determines what Presto content is visible in SharePoint and what SharePoint content is visible in Presto. This also determines how many Presto Add-On for SharePoint licenses are required.

The simplest connection pattern is a single connection defined between one Presto 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 Presto Server M is configured in the SharePoint Central Administration:




A corresponding connection to SharePoint Central Administration is defined in Presto Server M. With this configuration, all site collections within the SharePoint farm can see and use mashups or apps hosted in Presto Server M using a single Presto Add-On for SharePoint license.

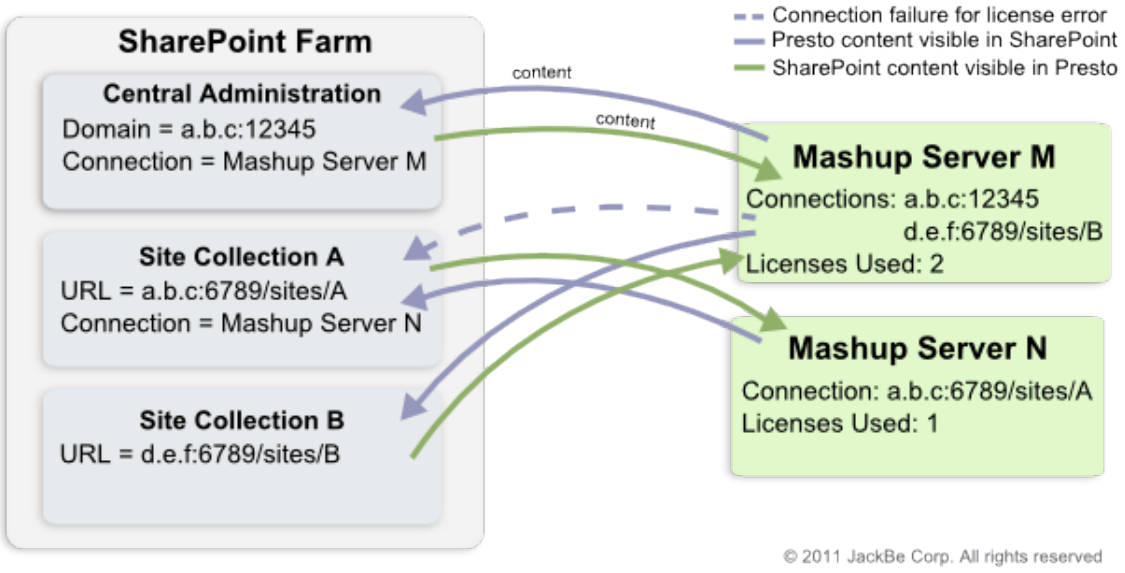
In Presto, however, only the SharePoint Lists defined in Central Administration are visible. To make SharePoint Lists from Site Collection A and Site Collection B visible in Presto, you would need to add separate connection configurations for each site collection in Presto Server M. Even with multiple site collections, however, only a single Presto 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 Presto Server. Presto Server N has a connection defined to the SharePoint Site Collection A. A corresponding connection is defined in Site Collection A to Presto Server N. This makes the mashup and apps in Presto 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 Presto users logged into Presto 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 Presto 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 Presto Server M:




- The connection to the SharePoint Central Adminisration in Presto Server M is required to allow mashups and apps in Presto Server M to be visible in the SharePoint farm. This works in conjunction with the connection configuration in the SharePoint Central Adminisration to Presto ServerM.

  This connection uses one license.

- A second connection to Site Collection B is required in Presto Server M to allow mashups and apps from Presto 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 Presto Server M from Central Adminisration configuration. But no mashups or apps from Presto 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 Presto 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 Presto Servers

and connections in each Presto Server to the relevant SharePoint site collections. For example:

| SharePoint Site Collections | Presto Servers |
|---|---|
| Site Collection A has connection configuration to:<br>■ Presto Server M<br>■ Presto Server N | Presto Server M has connections configuration to:<br>■ Site Collection A<br>■ Site Collection B |
| Site Collection B has connection configuration to Presto Server M. | Presto Server N has connection configuration to Site Collection A. |

## Add SharePoint Connections to the Presto Server

You must add connection information for SharePoint to the Presto Server*before* you can add connection information for the Presto Server to SharePoint. To determine what connections you should configure, see "Connection Patterns Between Presto Servers and SharePoint" on page 109 for examples and more information.

Each SharePoint connection in the Presto 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 Presto attributes. Users can also define their own credentials for a SharePoint connections, which create Presto 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 Presto 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 Presto 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 Presto Add-On for SharePoint, this is also the SharePoint server that the Presto Server works with to validate SSO tokens when users in SharePoint connect to Presto.

  **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 Presto Server.

- **Authentication Type** = the authentication protocol between the Presto 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 Presto Add-On for SharePoint.
- **Service Account Username** = the user name for the global credentials that the Presto 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 Presto 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 Presto Server along with SSO tokens. This name identifies the SharePoint server that should validate SSO tokens for the Presto Server.

> **Note:** This field is used *only* for the built-in, token-based single sign-on solution for Presto Add-On for SharePoint. It applies only to requests from SharePoint to the Presto 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 Presto 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 Presto 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 Presto 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, Presto 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 Presto Analytics

By default Presto uses local memory for caching and the Presto Analytics In-Memory Stores that Presto Analytics creates. This uses BigMemory as a local client that is installed with Presto and requires only your Presto license. In specific cases, you must also install BigMemory Servers on one or more additional hosts and configure Presto and the Integrated MashZone Server to work with them. Presto requires BigMemory Servers with a BigMemory license to support:

- Significant, extensible amounts of memory, most commonly for very large datasets used with Presto 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 Presto through the Terracotta Management Console (TMC) to allow Presto to work with this data. Apama, for example, dynamically creates distributed stores for the Apama MemoryStore which Presto can connect to and query.

- Distributed caching when Presto is deployed in clusters.

  With clusters, some of the interal Presto caches must be distributed and managed by BigMemory Servers.

- MashZone feeds that use BigMemory connections as a data source.

See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for instructions on configuring Presto 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 Presto Analytics that need different settings than the defaults. For more information on these tasks, see:

- "Declare BigMemory Stores for Presto Analytics" on page 120
- "Manage Dynamic BigMemory Stores for Presto Analytics" on page 123.

See "Caching for the Presto Server" on page 115 for an overview of Presto caching. For caching configuration, see "Configuring Mashable/Mashup Response Caching" on page 125 and "Distributed Caching for Presto Clusters" on page 116.

## Caching for the Presto Server

The Presto 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 116 and "Response Caching" on page 116 for details.

The Integrated MashZone Server also uses local memory for internal caches. See "Tune Memory/Caching for the Integrated MashZone Server" on page 149 for information on caching for the Integrated MashZone Server.

All Presto caches plus the Presto Analytics In-Memory Stores, can be distributed when Presto is deployed in clusters. See "Distributed Caching for Presto Clusters" on page 116 for an overview of distributed caching for Presto.

Caches for the the Integrated MashZone Server do not support distributed caching and must remain in local memory.

#### Artifact Caching

Artifact caching caches metadata for mashables, mashups and apps when they are run in Presto Hub, the AppDepot, the Presto 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 Presto 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 Presto.

#### 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 125 for instructions.

#### Distributed Caching for Presto Clusters

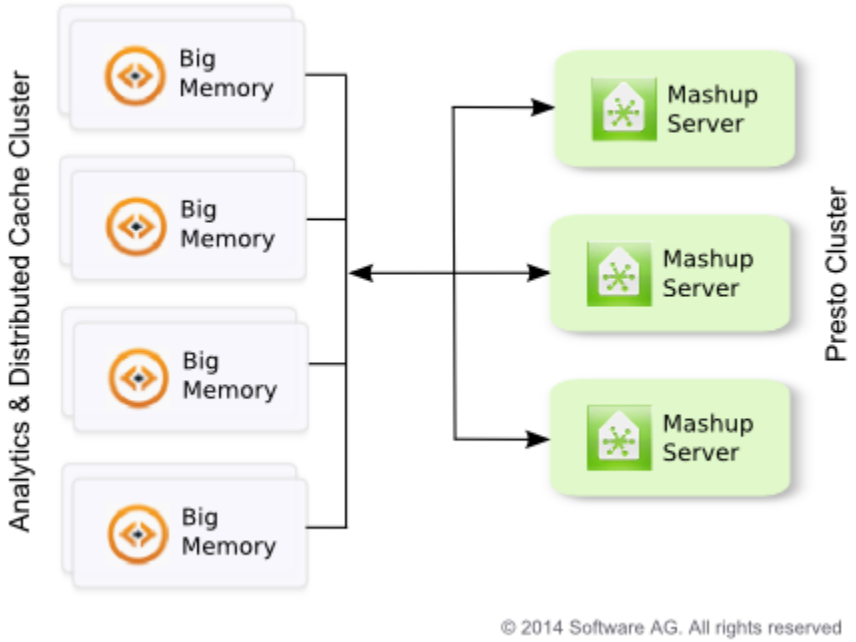
When Presto is deployed in clusters, both artifact caching and the Presto Analytics In-Memory Stores*must* be distributed to maintain cache integrity. Response caching, however, can be left in local memory for each Presto Server or it can be distributed.

In many environments, local caching provides both good performance and acceptable cache integrity for response caching. Local caching is "eventually consistent", but can result in visible differences as cached responses are not guaranteed to be identical for different cluster members. For environments that cannot tolerate any loss of cache integrity, distributed response caching is recommended.

**Note:** Distributed caching is only available if you purchase and deploy BigMemory Servers.

Caches for the the Integrated MashZone Server do not support distributed caching and must remain in local memory.

You use BigMemory Servers to handle distributed caching for Presto. When this is combined with Presto clusters, the high-level architecture looks like this:




With BigMemory Servers, data for both the Presto Analytics In-Memory Stores and most Presto caches can use the total off-heap memory configured for the cluster plus any heap and off-heap memory configured for the Presto 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 Presto Caching and In-Memory Stores" on page 117 for set up instructions.

## Configure BigMemory Servers for Presto Caching and In-Memory Stores

You can configure Presto 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 Presto Analytics" on page 114 for more information on features that require BigMemory servers.

### To configure BigMemory Servers for Presto

1. Copy your license for BigMemory to Presto and update Presto startup scripts:
   a. Copy the license file terracotta.key to the *presto-install* /apache-tomee-jaxrs/conf folder.

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

b. Add this Java system property to the Presto startup script :

```
-Dcom.tc.productkey.path=presto-install/apache-tomee-jaxrs/conf/
terracotta.key
```

Add this property to either:

- *presto-install* /apache-tomee-jaxrs/bin/setenv.bat file for Windows systems, or
- *presto-install* /apache-tomee-jaxrs/bin/setenv.sh file for Linux, OS/X or UNIX systems.

**Note:** If Presto is deployed in a cluster, you must update the startup scripts for every cluster member.

2. Edit the ehcache.xml file for Presto in a text editor of your choice. The location depends on your deployed environment:

| | |
|---|---|
| Only One Presto Server | *presto-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/ehcache.xml |
| A Presto Cluster and Shared External Configuration Folder | *presto-config* /ehcache.xml<br>For more information in a clustered environment, see "Setting Up an External Presto Configuration Folder" on page 258. |
| A Presto Cluster With No Shared Configuration | You must update *presto-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/ehcache.xml for each cluster member. |

3. Find the line in ehcache.xml with `<terracottaConfig>` that is commented out like this:

```
<!-- <terracottaConfig url="localhost:9510" /> -->
```

Remove the comment markers and change the `url` attribute to the host (or IP address) and port for the BigMemory server(s) you installed. For example:

```
<terracottaConfig url="tcHost1:9510" />
```

**Note:** There are several ways to identify one or more BigMemory servers for Presto. 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 Presto Analytics In-Memory Stores
- `SEARCH_RESULTS_CACHE` = one part of the Presto Artifact cache.
- `SERVICES_BY_ID_CACHE` = = one part of the Presto Artifact cache.

- `SERVICE_RESPONSE_CACHE` = the Presto 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 Presto 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 Presto cluster member.

6. Optionally, update configuration for dynamic In-Memory Stores that Presto Analytics creates to work with BigMemory servers:
   a. Edit the dynamiccache.xml file in these locations (based on your deployed environment):

| | |
|---|---|
| Only One Presto Server | *presto-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/dynamiccache.xml |
| A Presto Cluster and Shared External Configuration Folder | *presto-config* /dynamiccache.xml<br>For more information in a clustered environment, see "Setting Up an External Presto Configuration Folder" on page 258. |
| A Presto Cluster With No Shared Configuration | You must update *presto-install* /apache-tomee-jaxrs/webapps/presto/WEB-INF/classes/dynamiccache.xml for each cluster member. |

   b. Add a `<terracotta>` element inside the `<cache>` element.
   c. Provide URLs or other connection information as needed.

      For more information on configuration for the `<terracotta>` element, see BigMemory documentation.

   d. Save your changes.
7. Optionally, update configuration for any declared In-Memory Stores to work with these BigMemory server(s):
   a. In the configuration files for your declared In-Memory Stores, add a `<terracotta>` element inside the corresponding `<cache>` element.
   b. Provide URLs or other connection information as needed.

For more information on the `<terracotta>` element, see BigMemory documentation.

c. Save changes to these files.

d. Upload these configuration changes. See "Modify a Declared In-Memory Store" on page 122 for instructions.

8. Start BigMemory Server(s).
9. If needed, adjust memory configuration for the local Presto host. See "Memory Configuration for the Presto Server" on page 81 for instructions.
10. Restart Presto. See "Start and Stop the Presto Server" on page 20 for instructions.

## Declare BigMemory Stores for Presto Analytics

Declaring In-Memory Stores for Presto Analytics allows Presto 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 Presto Analytics.

**Note:** You can also define connections to in-memory stores that are created dynamically by external systems. See "Manage Dynamic BigMemory Stores for Presto Analytics" on page 123 for more information.

See "Declare a New In-Memory Store" on page 120, "Modify a Declared In-Memory Store" on page 122 and "View Details for Declared In-Memory Stores" on page 123 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 Presto.

   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 Presto 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 *presto-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 Presto, 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:** Presto 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 Presto supports the native Apama RowValue format. Presto 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 *presto-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 Presto is *presto-install* /apache-tomee-jaxrs/pretso/WEB-INF/lib. If Presto is deployed in a cluster, however, this location may be a separate external folder. For more information, see "Setting Up an External Presto Configuration Folder" on page 258.

3. Restart the Presto Server. For instructions, see "Start and Stop the Presto Server" on page 20.
4. Click  Admin Console in the Presto 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 Presto 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 Presto 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 Presto to this resource.

    **Note:** Administrators can also manage resources files in the Admin Console. See "Manage Files for Presto Features or Artifacts" on page 208 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 Presto in staging or production environments.

2. Click  Admin Console in the Presto 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 120 for instructions.

### View Details for Declared In-Memory Stores

To see configuration information for declared In-Memory Stores:

- Click  Admin Console in the Presto 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 Presto 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 Presto Analytics. For in-memory stores that are created dynamically by other systems, Presto 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 Presto Analytics" on page 120 for more information.

For information on the requirements for in-memory stores that act as dynamic external stores for Presto Analytics. For instructions on adding and managing external dynamic

store configuration, see "Add an External Dynamic In-Memory Store Connection" on page 124 and "Delete External Dynamic In-Memory Store Connections" on page 125.

## 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 Presto.

2. Copy the JAR file containing the classes used as search attributes to extract data from the dataset in this store to *presto-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 the destination folder in Presto is *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/lib. If Presto is deployed in a cluster, however, this location may be a separate external folder. For more information, see "Setting Up an External Presto Configuration Folder" on page 258.

3. Restart the Presto Server. For instructions, see "Start and Stop the Presto Server" on page 20.
4. Click  Admin Console in the Presto 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 Presto 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 Presto Server has a local Response Cache that is disabled by default. You must have Presto administrator permissions to enable response caching.

When Presto is deployed in clusters, you can continue to use local response caching or you can use distributed caching for the cluster. Distributed caching requires a cluster or server array of Terracotta BigMemory. You can install this cluster and then configure configure distributed caching for responses. See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for instructions.

### Enable and Configure Response Caching

For response caching to work, you must complete these steps:

1. Enable response caching for the Presto Server.

   Response caching configuration for individual mashables or mashups is ignored until a Presto administrator enables the Response Cache for the Presto Server.

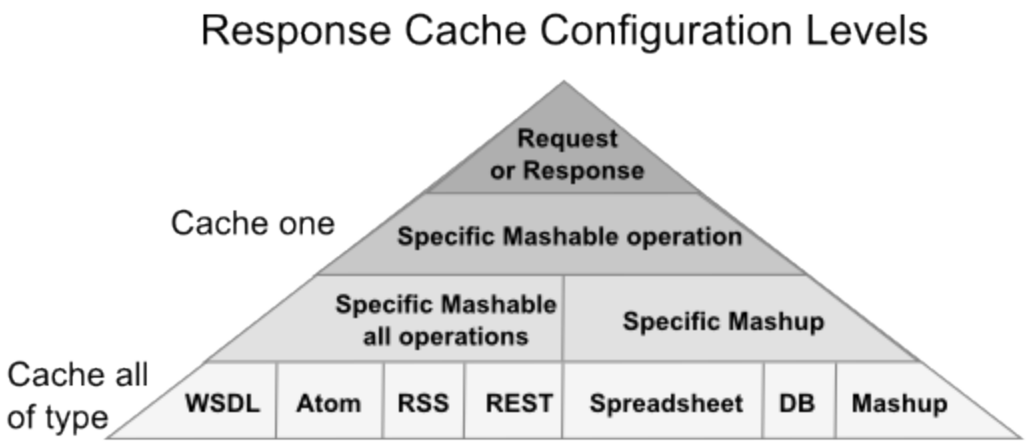
   a. Click  Admin Console in the Presto Hub main menu.

   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 129 for an illustration of the effects of caching configuration.

If you want to cache responses in most cases, it is best to enable caching and configure the expiration for entire types of mashables or mashups.Then disable or change expirations for the exceptions individually. See "Cache Responses by Default and Disable Exceptions" on page 126 for instructions.

If you only need to cache responses for specific mashables or mashups, it is best to leave caching disabled for all mashable types and mashups and enable caching only for specific exceptions. See "Cache Responses for Exceptions Only" on page 128 for instructions.

You can also override cache settings for individual requests or responses. See "Controlling Response Cache Entries Dynamically" on page 128 for more information.

## Cache Responses by Default and Disable Exceptions

This pattern enables response caching for all mashables of one or more types or for all mashups and disables caching only for specific mashables or mashups. You can also use this pattern to enable response caching for individual mashables and disable caching for specific operations.

**Note:** Only Presto administrators can configure response caching for all mashables of one type or all mashups. Only owners or Presto 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 Presto Hub main menu.

b. Expand the **Platform Features** section and click **Caching**.

c. If desired, change the **Default Maximum Age** for response cache entries. This defaults to 5 minutes. Enter a number and/or change the time interval.

d. Set the caching option for a mashable type or for mashups to enable caching for all mashables of that type or for all mashups.

   For example, to enable caching for all mashups, set the **Enable caching for Mashups** option.

e. If desired, set the maximum age for response cache entries for all mashables of this type or for all mashups. Enter a number or change the time interval.

   This default overrides the global default expiration period. Mashables or mashups can inheirit this expiration period or override it.

f. Enable caching for mashups or other mashable types as desired.

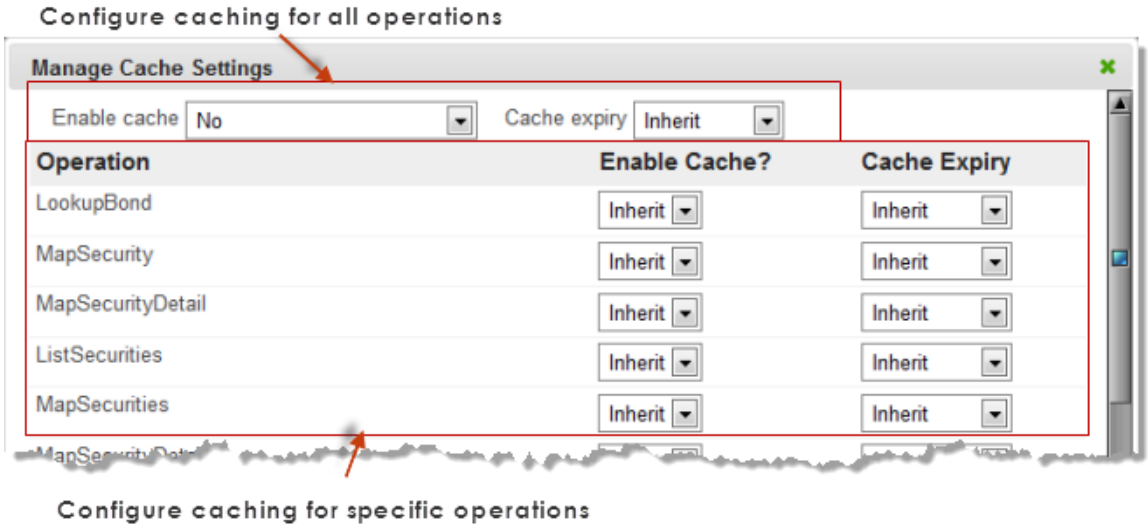
g. Click **Save**.

2. For mashables or mashups that should *not* have responses cached or that should use a different expiration period:

   a. Use search or browse to find the mashable or mashup you want and open its artifact page.

   b. Click **Manage >** **Cache**.

      You can control response caching for all operations for a mashable, for mashups or for specific operations of a mashable as shown in the following example:




   c. To turn response caching off for all operations for this mashable or mashup, set **Enable cache** to `No`.

d. To leave response caching on but change the expiration period for cache entries for all operations of this mashable or mashup, change the **Cache expiry** from `Inherit` to a specific value.

e. If needed, select specific operations and turn response caching on or update their expiration periods in the corresponding operation fields.

f. Click **Close**.

You can also update cache settings to multiple artifacts using the bulk update feature.

## 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 Presto 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 Presto Connect or the Presto 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 Presto 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 Presto 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.<br><br>**Note:** If no caching configuration had been set in step 1, no response caching would occur after this |

| | Event | Cache Entry |
|---|---|---|
| | | step as both are required for caching to occur. |
| 3. | A Presto administrator turns caching on for all RSS and Atom mashables, but does not configure an expiration period.<br><br>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.<br><br>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.<br><br>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.<br><br>UserB uses this app. Three minutes later, UserC also uses this app. | The mashup is run and caches its response when UserB runs the associated app. The mashup also runs the Yahoo Financial mashable and that response is also cached.<br><br>When UserC runs the app three minutes later, the cache entry for Yahoo Financial has expired. The app, however, simply uses the cached mashup response which is based on Yahoo Financial results from three minutes before that may now be considered stale. |
| 6. | A Presto 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 | When users work with the REST mashable, or any mashup or app |

| | Event | Cache Entry |
|---|---|---|
| | 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. | based on this mashable, responses will be cached except for those operations where caching has been disabled. |
| 8. | A Presto developer creates a mashup that uses several Atom mashables. In the mashup he adds the HTTP Cache-Control header with a value of `no-cache` to the response of each of the Atom mashable invocations. | When this mashup is used, each Atom mashable will be invoked every single time with no response caching because of the HTTP response headers. |

# Manage Data Sources and Drivers

Data sources combine the connection and driver information needed to work with both the Presto Repository and with other databases. Datasources can use either JDBC connections or a JNDI connection pool. Once you have created a data source, you and other users can use it as the source for any number of mashables or in mashups using EMML.

See "Add a Data Source" on page 131, "Edit, Test or Remove Data Sources" on page 133 and "Add or Manage JDBC Drivers" on page 134 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.

**To add a data source**

1. Click  Admin Console in the Presto Hub main menu.
2. Expand the **JDBC Configuration** tab.
3. If needed, add the JDBC driver for this database to Presto. See "Add or Manage JDBC Drivers" on page 134 for instructions.
4. Click **Data Sources** to see a list of existing data sources.

   Initially, this lists the data source for the Presto Repository and for the Snapshots repository.
5. Click **Add new data source** to create a new data source.

6. Enter a **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.
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.

   Typically, you must specify at least a user name although password may be optional based on this database.

10. Optionally, set connection pooling options for this datasource:
    - **Initial size** = the initial number of connections to create when the pool for this datasource starts up. This defaults to 0.
    - **Max active** = the maximum number of connections that can be allocated at one time for this datasource. This defaults to 8. Set this to -1 to remove all limits.
    - **Max 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.
    - **Max idle** = the maximum number of connections that can be idle without connections being released for this datasource. Defaults to 8. Set this to -1 to prevent any connections being released.
    - **Min idle** = the minimum number of idle connections that can exist before new connections are added to the pool for this datasource. 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 run millis** = 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 eviction run** = the number of connections to test during any idle test run for this datasource. This defaults to three.
- **Min evictable idle time millis** = the minimum number of milliseconds that a connection can be idle before being tested for eviction. This defaults to 30 minutes.

**Note:** For more details on connection pooling properties, see Apache DBCP Documentation

11. Click **Save**.

## Edit, Test or Remove Data Sources

**To manage data sources**

1. Click  Admin Console in the Presto Hub main menu.
2. Expand the **JDBC Configuration** tab.
3. Click **Data Sources** to see a list of existing data sources.

   Initially, this lists the data source for the Presto Repository and for the Snapshots repository.
4. To edit a data source, click  **Edit** and change properties.

   See "Add a Data Source" on page 131 for information on specific data source properties.
5. To test the connection to a data source, click  **Test**.
6. To delete a data source, click  **Delete** on the line for that data source.

**Important:** Do *not* delete the data source for either the Presto Repository or the Snapshots repository.

## Add or Manage JDBC Drivers

**To manage JDBC drivers**

1. Click Admin Console in the Presto Hub main menu.
2. Expand the **JDBC Configuration** tab.
3. Click **JDBC Drivers**.

   A list of existing drivers displays. This initially contains just the driver for the default Presto Repository.
4. To remove a driver, click **Delete** on the line for that driver.

   **Important:** Do *not* delete the driver for the Presto Repository.
5. To add a new driver:
   a. Click **Add new JDBC driver**.
   b. Enter a **Name** for a new driver.
   c. Enter the Java **Class** name for this driver.
   d. Click **Browse** and find the JAR file for this driver.
   e. Click **Add this JDBC driver**.

# Configure the Default Operations Generated for Database Mashable

When users or Presto administrators register simple or custom mashables for databases, Presto 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. Presto 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 Presto 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. Presto 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 Presto 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 Presto 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 Presto. 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 Presto, 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 Presto 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 Presto Attributes

You use Presto 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.

Presto attributes can be defined for these contexts:

- **User Attributes**: are defined by each user as part of their profile in Presto and saved in the Presto User Repository.

You can also choose to expose attributes from the Presto User Repository as Presto user attributes. You can expose attributes from the default User Repository or from your LDAP Directory, if Presto is configured to use LDAP. See "Expose User Attributes from the User Repository in Presto" on page 139 for more information.

At runtime, the Presto Server first checks the Presto user attributes for the current user to resolve any references to Presto attributes used in a request. If there is no matching user attribute for the current user, the Presto Server uses the matching Presto 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 Presto 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 Presto Global Attributes" on page 138 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 Presto session attributes are defined in requests using a Presto 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 140 for more information.

## Manage Presto Global Attributes

Presto global attributes define:

- Default or common values that should be used for input parameters to mashables, mashups or apps.
- Default or shared values for Presto 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 Presto user attribute of the same name.
- Credentials or other configuration used in connections to SharePoint when your site has the Presto Add-On for SharePoint Add-On. For more information on global attributes used with SharePoint connections, see configuring "Configure Presto Connections to SharePoint" on page 109.
- Schemas for datasets used in RAQL queries in Presto Analytics.

**To manage Presto global attributes**

1. Click Admin Console in the Presto 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.

      Presto attributes can have any name, but they must be unique within a given Presto Repository. If a global attribute is to be used as a default for a Presto 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 Presto 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 Presto Server to apply these changes. See "Start and Stop the Presto Server" on page 20 for instructions.

## Expose User Attributes from the User Repository in Presto

You can access specific user attributes from the User Repository when you run apps, mashables or mashup using Presto user attributes. Typically, you use Presto attributes (user, global or session) as tokens for the values for input parameters. The Presto Server resolves these tokens when the artifact is run.

The specific attributes from your user repository that are visible as Presto user attributes depends on how you have configured the User Repository in Presto:

| | |
|---|---|
| Default User Repository | All repository attributes are accessible. This includes:<br>■ `firstName`<br>■ `lastName`<br>■ `email`<br>■ `username`<br>■ `password` |

| | |
|---|---|
| LDAP Directory | The LDAP attributes that are visible as Presto user attributes is determined by the configuration you set when you integrate LDAP with Presto. By default, this includes:<br>■ `uid`<br>■ `givenName`<br>■ `sn`<br>■ `mail`<br>■ `cn`<br>■ `postalCode`<br>See "Integrate Your LDAP Directory with Presto" on page 40 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 Presto 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 140 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 Presto 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 Presto 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 Presto for artifacts of that type. It does *not* remove any values assigned to artifacts in Presto 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 Presto Server:

**To enable/disable mashup features**

1. Click Admin Console in the Presto Hub main menu.
2. Expand the **Platform Features** section and click **Mashup Engine**.
3. Set or clear the following properties:
   - *Enable Scripting* = this feature is the <script> statement in EMML which allows mashups to call JavaScript or JRuby scripts. This also provides direct access to Java language classes.

     This feature is enabled by default. Scripting can have security implications and also has a performance impact. Clear this option to disable scripting.
   - *Enable Direct SQL* = this feature is the <sql> and <sqlUpdate> statements in EMML. These statements allow mashups to issue raw SQL statements to databases which can have security implications.

     This feature is enabled by default. Clear this option to disable direct SQL statements in mashups.
   - *Enable EMML Profiling* = this option determines whether performance profiling for mashups is enabled. Performance profiling allows developers to see how long processing took for each statement in their mashups when they test them. Profiling does, however, have a performance impact.

Performance profiling is disabled by default. Set this option to enable performance profiling.

- *Enable direct invocation* = this feature is the <directinvoke> statement in EMML. This statement allows mashups to invoke web services or other information sources from any URL without authentication or authorization by Presto.

  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 Presto 3.1 and above.

When mashable information sources or mashups are invoked in Presto, the Presto 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 Presto Server defines those standard HTTP response headers from mashable or mashup responses that should automatically be included in Presto 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 Presto 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 Presto 3.1.

Presto 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 Presto Hub main menu.
2. Expand the Platform Features section and click **HTTP Processing**.
3. In the HTTP Request Processing section:
   - Enter a number of milliseconds in **Connection Timeout** to define the timeout period for connections to be successfully made with mashable information sources.
   - Enter a number of milliseconds in **Socket Timeout** to define the timeout period for a response to be received from mashable information sources.
4. Click **Save settings** (in the HTTP Request Processing section).

## Enable or Disable the Snapshot Feature

The snapshots feature allows users to save the specific results when a mashable or mashup is run and use this data in mashups or apps. Users or administrators can also schedule snapshot jobs to take snapshot automatically.

This feature is disabled by default when you install Presto.

**To disable/enable the snapshot feature**

1. Click Admin Console in the Presto Hub main menu.
2. Expand the **Platform Features** section and click **Snapshots**.
3. Set the **Enable Snapshots** option.

# Set Web Feed Normalization

By default, Presto 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 Presto 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 Presto 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 Presto 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 Presto Server.

**To add a configuration property**

1. In any simple text editor, open the presto.config file in the *presto-config* folder.

   **Note:** The *presto-config* folder may be an external folder outside the Presto Server or it may be in the default locations. See "Setting Up an External Presto

Configuration Folder" on page 258 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 Presto Server. See "Start and Stop the Presto Server" on page 20 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 Presto 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 Presto Features or Artifacts" on page 208 for more information.

# 5 Integrated MashZone Server Configuration and Administration

Presto uses the Integrated MashZone Server to manage and process MashZone feeds that you import into Presto or that users create in Presto, 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 Presto mashables and mashups in that they access information from a variety of sources. For Presto, MashZone feeds also allow you to access data from Optimize or Process Performance Manager.

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

- Move the MashZone Repository to a robust solution. See "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 for instructions.
- Make the MashZone Feed Editor available in the Presto Hub. See "Enable the MashZone Feed Editor" on page 156.
- Make the MashZone Administration available in the Presto Hub. See "Enable the MashZone Administration" on page 157.
- Set basic configuration for the Integrated MashZone Server, such as proxy server information. See "Basic Settings for the Integrated MashZone Server" on page 148 for instructions.
- "Tune Memory/Caching for the Integrated MashZone Server" on page 149
- Import existing MashZone feeds into Presto.

  See "Import or Export MashZone Feeds" on page 153 for instructions.
- Create or manage database connections used in MashZone feeds. See "Add Database Connections for the Integrated MashZone Server" on page 155 or "Manage Database Connections for the Integrated MashZone Server" on page 156 for instructions.
- "Event Service Configuration and Administration" on page 159 The Integrated *Event Service* allows Presto to create and handle *event mashables* that connect to the Event Bus andApama , subscribe to specific event types and then use data from published events as an information source.
- Integrate with Process Performance Manager (PPM). This configuration allows users to work with the PPM Chart app to find and include PPM charts in workspace apps or to define data sources for MashZone feeds from PPM.

  See "Process Performance Manager (PPM) Integration" on page 189 for more information and links.

## Basic Settings for the Integrated MashZone Server

Once you have defined the connection to the Integrated MashZone Server, you can complete basic settings that allow you to:

- Create New MashZone Resource Directories

Data sources used in MashZone feeds that are file based, such as Excel spreadsheets, CSV files or XML files, must be stored in resource directories known to the Integrated MashZone Server. Resource directories also allow you to restrict access to data sources to specific users or user groups.

- Manage MashZone Resource Directories

  You can update, delete, share, export and import resource directories.

- Configure Proxy Information for the Integrated MashZone Server
- Enable the Geocoding Operator for MashZone Feeds

  This operator uses the Google Maps API. To enable the operator, you must accept license terms and optionally provide an API key.

## Tune Memory/Caching for the Integrated MashZone Server

Presto, the Integrated MashZone Server and the Event Service share the local Java heap memory. Heap memory is also used for internal caches and PrestoIn-Memory Stores used in Presto Analytics. Presto can also be configured to use off-heap memory if you have installed BigMemory Servers. For more information, see "Memory Configuration for the Presto Server" on page 81 .

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 Presto, the PrestoIn-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 Presto, 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 Presto or PrestoIn-Memory Stores into account and thus may not be suitable in some circumstances.

# Tune MashZone Memory and Cache Configuration Manually

**To manually update memory and cache configuration**

1. Update Java Heap Settings
2. Update Cache Memory Settings
3. Update MashZone ThreadSize Properties.
4. Then restart the Presto Server to apply this change. See "Start and Stop the Presto Server" on page 20 for instructions.

### Update Java Heap Settings

**To update the size of the Java Heap**

1. In a text editor of your choice, open the setenv.bat or setenv.sh file, based on your operating system, in the *presto-install* /apache-tomee-jaxrs/bin/ folder.
2. At a minimum, update the `-Xmx` value to tune the maximum size of heap. See the preset suggestions in Automatically Tune MashZone Memory and Cache Configuration as starting points and Memory Configuration for the Presto Server for suggestions for Presto.
3. Save your changes.

### Update Cache Memory Settings

1. In the text editor of your choice, open the ehcache.xml file in the *web-apps-home* / mashzone/WEB-INF/classes folder.
2. Update the `maxBytesLocalHeap` value on the `<cache>` elements with the following names:
   - `RESULT_FEED_BASE`
   - `RESULT_FEED_TOP`
   - `RESULT_FEED_DEBUG`

   See the preset suggestions in Automatically Tune MashZone Memory and Cache Configuration as starting points.
3. Save your changes

### Update MashZone ThreadSize Properties

1. In the text editor of your choice, open the mashzone.properties file in the *web-apps-home* /mashzone/WEB-INF folder.
2. Update the following properties:
   - `calculation.threadpool.coresize`

- `calculation.threadpool.maxsize`

  See the preset suggestions in Automatically Tune MashZone Memory and Cache Configuration as starting points.

3. Save your changes

## Automatically Tune MashZone Memory and Cache Configuration

This uses a simple script to automatically update memory and cache configuration based on preset sizes. These preset values, however, do not take any memory requirements for Presto or PrestoIn-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:<br>- 64 bit<br>- 2 Cores<br>- 4G of memory | 1G | 4 | 4 | - Base = 125M<br>- Top = 100M<br>- Debug =25M |
| *M*: a medium application on a host with:<br>- 64 bit<br>- 4 Cores<br>- 16G of memory | 8G | 8 | 12 | - Base = 1G<br>- Top = 800M<br>- Debug =200M |
| *L*: a large application on a host with:<br>- 64 bit<br>- 8 Cores<br>- 64G of memory | 16G | 16 | 24 | - Base = 2G<br>- Top = 1.6G<br>- Debug = 400M |

2. Open a command or terminal window and move to the *presto-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 Presto and the Integrated MashZone Server. See "Memory Configuration for the Presto Server" on page 81 for instructions.

## Create New MashZone Resource Directories

To work with data sources in MashZone feeds that are file-based, such as Excel spreadsheets, CSV files or XML files, you must store the files in a *resource directory* that the Integrated MashZone Server knows. This can be the default resource directory:

*presto-install* /mashzone/data/resources

Or it can be any subdirectory of the default.

You can also use resource directories to control access to data source files to specific users or groups.

1. Click Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. See the MashZone documentation for the remaining steps.

## Manage MashZone Resource Directories

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

1. Click Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. Consult the MashZone documentation to perform the following:
   - Edit a Resource Directory
   - Delete a Resource Directory

- Share Resource Directories to determine which users and groups can work with these data sources.
- Import Resource Directories
- Export Resource Directories

## Configure Proxy Information for the Integrated MashZone Server

If a proxy server is used to access data sources for MashZone feeds, you must add configuration information for the proxy server.

1. Click  Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. See the information on Set proxy server settings in the MashZone documentation for the remaining steps.

## Enable the Geocoding Operator for MashZone Feeds

This operator for MashZone feeds uses The Google Maps APIs. To use this operator, you must accept the Google license terms.

You may also need to enable unrestricted access with a Google Maps API Premier key to support cumulative transactions greater than 2,500 per day for the MashZone feeds that use this operator.

1. Click  Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Server Settings**.
4. See the information on entering Google Maps key in the MashZone documentation for the remaining steps.

## Import or Export MashZone Feeds

You can import existing MashZone feeds from your existing MashZone Servers into Presto to use them to create apps or workspace apps. You can also export any or all MashZone feeds registered in Presto and the Integrated MashZone Server.

## Import Existing MashZone Feeds

If you have data feeds defined in an existing MashZone Server, you can import them into Presto to use them in apps or workspace apps.

1. Export the MashZone data feeds that you want from your existing MashZone Server and *omit* permissions. See MashZone help for instructions on completing this step.

   **Important:** Do not export MashZone feeds that use connections to Terracotta BigMemory caches as a data source. These connections are not supported by the Integrated MashZone Server for Presto.

   Instead, connect to BigMemory caches using Presto Analytics.

   This creates an MZP export file (*.mzp) in the default export/import folder for your existing MashZone Server.

2. Copy the MZP export file you created in the previous step to the *presto-install* / mashzone/data/importexport folder.
3. Click  Admin Console in the Presto Hub main menu.
4. Click **MashZone** to expand this section of the Administration menu.
5. Click **Import/Export**.
6. To import all MashZone feeds defined in any MZP export files in the *presto-install* / mashzone/data/importexport folder, click **Import All** and confirm this choice.
   a. A validation status message is displayed. Click **OK**.

      The import begins. You become the owner for all the MashZone feed(s) that are imported.

   b. An import status message is displayed. Click **OK**.
7. To import specific MashZone feeds from the MZP export files in the *presto-install* / mashzone/data/importexport folder, click **Import** and:
   a. If needed, change the **File Format** to see a list of specific export files.
   b. Select the MZP export file containing the MashZone feeds you want to import.
   c. Click **OK**.
   d. A validation status message is displayed. Click **OK**.

      The import begins. You become the owner for these MashZone feed(s).

   e. An import status message is displayed. Click **OK**.

You can now find the MashZone feeds in Presto search results, open them, add views, create apps or add them to workspace apps. To allow other Presto users to work with these feeds, you must also grant them permission to run these feeds.

## Export MashZone Feeds from Presto

You can export any or all MashZone feeds that you have imported into Presto or created in Presto to migrate them to another Presto instance.

1. Click Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Import/Export** and select the **Export** tab.
4. To export specific MashZone feeds:
   a. Find and select the specific MashZone feeds you want to export from the list.
   b. Click **Export Selection**.

      This creates an MZP export file in the *presto-install* /mashzone/data/importexport folder with a name in the form F*_feed-name _date-and-time* .mzp that you can use to migrate these MashZone feeds.
5. To export all MashZone feeds:
   a. Click **Export All**.
   b. When prompted, click **Export All** to confirm this export.

      This creates an MZP export file in the *presto-install* /mashzone/data/importexport folder with a name in the form F*_first-feed-name _date-and-time* .mzp that you can use to migrate these MashZone feeds.

# Add Database Connections for the Integrated MashZone Server

To allow users to work with database data sources in MashZone feeds, you must add database connections to the Integrated MashZone Server:

1. First, you Add a Database Driver to MashZone.
2. Then Add a Database Connection to MashZone.

## Add a Database Driver to MashZone

For information on MashZone requirements for database drivers, see the information on MashZone JDBC Driver Requirements in the MashZone documentation.

1. Obtain the JAR file with the appropriate JDBC driver for the database you want to connect to. See your system administrator or documentation for this database for more information.
2. Move the JDBC jar file to the *presto-install* /mashzone/data/jdbcdriver folder.

3. Restart the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

   Restarting the Presto Server also restarts the Integrated MashZone Server, allowing MashZone to discover the new database driver and appropriate classes.

### Add a Database Connection to MashZone

1. Click Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Database connections**.
4. See Create Database Connection in the MashZone documentation for the remaining steps.

## Manage Database Connections for the Integrated MashZone Server

1. Click Admin Console in the Presto Hub main menu.
2. Click **MashZone** to expand this section of the Administration menu.
3. Click **Database connections**.
4. See the following information in the MashZone doumentation for the remaining steps to edit, delete, export or import database connections:
   - Edit database connection
   - Delete database connection
   - Export database connection
   - Import database connection

## Enable the MashZone Feed Editor

By default, users cannot create new MashZone feeds in the Feed Editor. This editor is hidden from users in the Presto Hub main menu.

You can make the Feed Editor visible to users in the **Mashups** menu in the Presto Hub main menu to allow users to create new MashZone feeds.

Procedure

1. Open the file presto.conf in your text editor.

   *presto-install* \apache-tomee-jaxrs\webapps\presto\WEB-INF\classes\presto.conf

2. Configure the `mashzone.feededitor.disabled` parameter.
   - Set the parameter `=false` to enable the MashZone Feed Editor.
     `mashzone.feededitor.disabled=false`
   - Set the parameter `=true` to disable the MashZone Feed Editor.
     `mashzone.feededitor.disabled=true`
3. Save your changes.
4. Reload the Presto bowser tab.

This makes the **MashZone Feed Editor** in the Presto Hub **Mashups** menu visible.

## Enable the MashZone Administration

By default, users cannot use the MashZone Administration. This Administration is hidden from users in the Presto Hub main menu.

You can make the MashZone Administration visible to users in the **Mashups** menu in the Presto Hub **Admin Console** to allow users to manage specific MashZone features.

Procedure

1. Open the file presto.conf in your text editor.
   *presto-install* \apache-tomee-jaxrs\webapps\presto\WEB-INF\classes\presto.conf
2. Configure the `mashzone.administration.disabled` parameter.
   - Set the parameter `=false` to enable the MashZone Feed Editor.
     `mashzone.administration.disabled=false`
   - Set the parameter `=true` to disable the MashZone Feed Editor.
     `mashzone.administration.disabled=true`
3. Save your changes.
4. Reload the Presto bowser tab.

This makes the MashZone Administration in the Presto Hub **Admin Console** visible.

To open the MashZone Administration click  Admin Console in the Presto Hub main menu. Then click **MashZone** to expand this section of the Administration menu.

# 6 Event Service Configuration and Administration

### About the Event Service and Event Data

The Event Service that is integrated with Presto allows Presto to connect to the Event Bus for Software AG and subscribe to events published by other Software AG applications. In most cases, the system acting as the Event Bus is Universal Messaging or the webMethods Broker using a Messaging Service, although other systems and transports may serve as the Event Bus.

The Event Bus handles events published by *event producers* which may be a variety of Software AG applications. It routes events as they are published to *event consumers*, such as Presto, who have subscribed to specific *event types*, also known as *channels*.

Presto 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 Presto, each Event Bus or Apama subscription feeds an *event source* which is managed by the Event Service. Event sources receive events for subscriptions, store them in memory and act as the data source for the corresponding EDA or Apama event.

### Use Events as Information Sources

To use events as information sources, you must create EDA or Apama event sources. Unlike other sources, only Presto administrators can create EDA and Apama event sources for other users to work with.

To create these sources, administrators must:

1. For EDA event source "Identify the Event Type Store directory" on page 161. This defines the types of events that are available for subscriptions from the EDA event source.
2. Create event sources of the Event Type Store and/or Apama as needed. Presto retrieves event data from these event sources when users run the corresponding EDA or Apama event source. If views for the event source are real-time views, events are pushed to the view automatically.
   - Create EDA Event Sources
   - Create Apama Event Sources
3. "Start or Stop an Event Source" on page 186

You can also Manage Apama Event Sources and Manage EDA Event Sources.

# Manage EDA Event Sources

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

1. Click  Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.

3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Select further steps:
   - "Identify the Event Type Store directory" on page 161
   - "Create EDA Event Sources" on page 161
   - "Edit EDA Event Sources" on page 167
   - "Duplicate EDA Event Sources" on page 172
   - "Delete EDA Event Sources" on page 172
   - "Import EDA Event Sources" on page 173
   - "Export EDA Event Sources" on page 173

## 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 Presto. To begin configuration for event subscriptions and their associated event sources, you must first identify the Event Type Store directory.

**To identiy the Event Type Store directory:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click **Event Type Store directory**.
6. Enter the path to the local Event Type Store in the **Event Type Store directory** field.
7. Click **Save**.

Your changes are applied.

## Create EDA Event Sources

"You have identified the Event Type directory." on page 161

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 Presto.

**To create an EDA Event Source:**

1. Click Admin Console in the Presto Hub main menu.

2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click **Create EDA Event Source**.
6. Set the properties for this event source. See table "EDA Event Source properties" on page 162 below.
7. 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 Presto Server starts. Clear this option if you need to manually control startup for this event source. |
| EDA URL | yes | URL to Universal Messaging Service:<br>nsp://<server>:<port><br>Example: nsp://localhost:9000] |
| Event type | yes | Whether events for this event source come from the Event Bus (an EDA connection) connection.<br>Select the type of the event this event source should subscribe to.<br>The XML schema files for these event types must exist in the Event Type Store directory. |
| Channel | | Leave this property blank, unless the channel the event source should subscribe to is *not* the default channel for the selected event type. If the channel is not the default, select the correct channel. |
| Filter predicates | | Enter a filter expression defining the events to be published to this event source.<br>Detailed information on filtering events is available in documentation for Universal Messaging or the |

| Property | Required | Description |
|---|---|---|
| | | webMethods Broker, based on which system is acting as the Event Bus in your environment. |
| Check validity | | Available only if **Strategy** is set to `Buffer`.<br>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.<br>■ 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.<br>■ The current time of the application is determined by the start time of the last received event.<br>■ 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:<br>■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events.<br>■ `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`.<br>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.<br><br>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.<br><br>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.<br><br>Default value: 1 (Max: 100.000).<br><br>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.<br><br>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.<br><br>This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached.<br><br>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).<br><br>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.<br><br>Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value.<br><br>Default value: 10 |

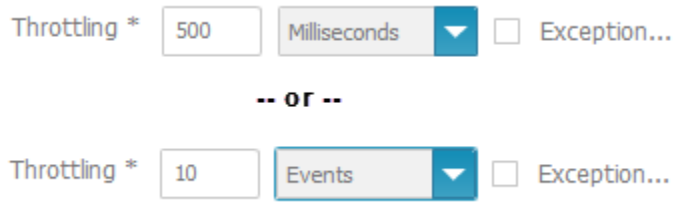
| Property | Required | Description |
|---|---|---|
| | | The product from **Max number of dimension values** and **Capacity per dimension value** must not be more than 100 000. |
| Event ID attribute | yes | Available only when **Strategy** is set to `DELTA`.<br>Select the attribute that identifies 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`.<br>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)<br>Default value: 10 |
| Memory model | | Determines where events are stored:<br>■ `Internal`: the default which stores events in local memory for this event source.<br>■ `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:<br>■ Change the number of milliseconds to control throttling.<br>■ 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.<br>See "example below" on page 166. |
| 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):<br>■ A number<br>■ `Milliseconds` or `Events` as the measurement for the exception criteria<br>See "example below" on page 166 |

**Simple and Hybrid Throttling Strategies**

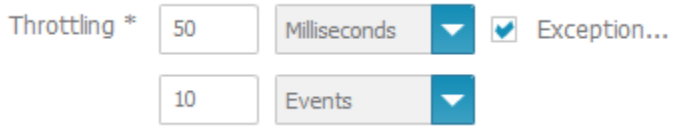
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events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

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:




Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:




With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.

- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.

On the EDA Event Source overview page you can click on the **Alias** to show a preview of the specific Event Source properties.

## Edit EDA Event Sources

You can edit already existing EDA Event Source.

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

**To edite an EDA Event Source:**

1. Click  Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the **Edit** icon to configure a specific EDA connection.
6. Set the properties for this event source:

**Table 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 Presto Server starts. Clear this option if you need to manually control startup for this event source. |
| EDA URL | yes | URL to Universal Messaging Service:<br>nsp://<server>:<port><br>Example: nsp://localhost:9000] |
| Event type | yes | Whether events for this event source come from the Event Bus (an EDA connection) connection.<br>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. |
| Channel | | Leave this property blank, unless the channel the event source should subscribe to is *not* the default channel for the selected event type. If the channel is not the default, select the correct channel. |
| Filter predicates | | Enter a filter expression defining the events to be published to this event source.<br>Detailed information on filtering events is available in documentation for Universal Messaging or the webMethods Broker, based on which system is acting as the Event Bus in your environment. |
| Check validity | | Available only if **Strategy** is set to `Buffer`.<br>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.<br>■ 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.<br>■ The current time of the application is determined by the start time of the last received event.<br>■ 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:<br>■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events. |

| Property | Required | Description |
|---|---|---|
| | | ■ 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.<br>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.<br>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.<br>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.<br>Default value: 1 (Max: 100.000).<br>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.<br>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.<br>This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached. |

| Property | Required | Description |
|---|---|---|
| | | 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).<br>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.<br>Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value.<br>Default value: 10<br>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`.<br>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`.<br>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)<br>Default value: 10 |
| Memory model | | Determines where events are stored:<br>- `Internal`: the default which stores events in local memory for this event source.<br>- `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. |

| Property | Required | Description |
|---|---|---|
| | | By default, event sources push event data every 500 milliseconds. You can:<br>■ Change the number of milliseconds to control throttling.<br>■ 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.<br>See "example below" on page 166. |
| Exception | | Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1):<br>■ A number<br>■ `Milliseconds` or `Events` as the measurement for the exception criteria<br>See "example below" on page 166 |

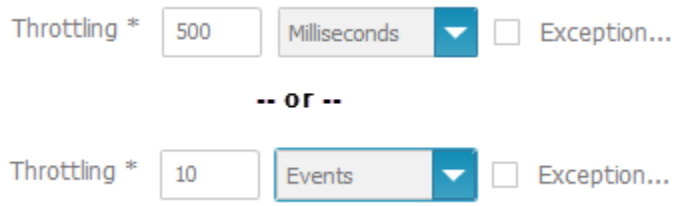
7. Click **Save**.

Your changes are applied.

#### Simple and Hybrid Throttling Strategies

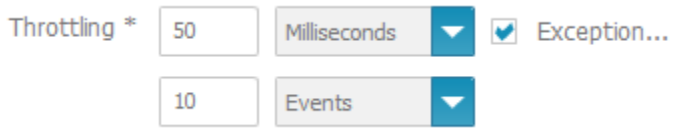
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events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

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:




Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:




With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.
- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.
- see

Create EDA Event Sources

## Duplicate EDA Event Sources

You can duplicate existing EDA Event Source.

**To duplicate an EDA Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the **Copy** icon to duplicate a specific EDA Event Source.

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

## Delete EDA Event Sources

You can delete existing EDA Event Source.

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

**To delete an EDA Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click the **Delete** icon to delete a specific EDA Event Source.

The selected EDA Event Source is deleted from the list.

## Import EDA Event Sources

You can import individual configurations of EDA Event Sources. You can import only EDA configurations that are stored in the importexport directory presto-install \mashzone\data\importexport or in one of the subfolders.

The files are saved as archive files, *.mzp. If an EDA Event Source with the same name already exists it cannot be imported. The existing Event Source must be deleted first.

**To import an EDA Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **EDA** tab.
5. Click **Import EDA Event Source**.
6. Select an EDA .mzp file from the list:.
7. Click **Import EDA Event Source**.

The selected EDA Event Source with all relevant settings is imported and displayed.

## Export EDA Event Sources

You can export the configurations of EDA Event Sources. The exported EDA Event Sources are stored directly in the importexport directory presto-install\mashzone\data \importexport. The EDA Event Sources are saved with the corresponding settings and shared as *.mzp archive file.

**To export an EDA Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.

4. Open the **EDA** tab.
5. Click the Export button to export an EDA Event Source.
6. Click **Close**.

The selected EDA Event Source is saved in the importexport directory as a mzp archive file.

# Manage Apama Event Sources

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

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Select further steps:
   - "Create Apama Event Sources" on page 174
   - "Edit Apama Event Sources" on page 179
   - "Duplicate Apama Event Sources" on page 184
   - "Delete Apama Event Sources" on page 185
   - "Import Apama Event Sources" on page 185
   - "Export Apama Event Sources" on page 186

## Create Apama Event Sources

Presto can work with events published from Apama through the Event Bus. In many cases, however, the events and data you need are defined in Apama*scenarios* which are not accessible through the Event Bus.

To work with Apama scenario events, you must create an Apama Event Source to receive scenario events.

**To create new Apama Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click **Create Apama Event Source**.

6. Set the properties for this event source. See table "Table 3" on page 175 below.
7. 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 Presto Server starts. Clear this option if you need to manually control startup for this event source. |
| **Apama** URL | yes | URL to the running **Apama** system ( local or remote) |
| **Apama** Scenario | yes | Click **Refresh** to update the list of **Apama** scenarios for the selected Apama Event Source. If the **Apama** URL is set to a valid **Apama** system, then it is possible to select a scenario this event source should subscribe to. |
| Strategy | yes | The strategy that this event source uses for saving and removing events published from the Event Bus. Valid strategies are:<br>■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events.<br>■ `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.<br>■ `PARTIAL_EVENTS` = Each event has a unique identifier defined by one or more key fields (see Key attributes). Events contain additional fields, but may not contain all fields possible for the event. Simply put, each event may contain partial data. |

| Property | Required | Description |
|---|---|---|
| | | The event source maintains a single row for each unique event key representing the current full status for that event. Events published by Apama scenarios update the fields in that event source row that are included in the event, leaving other existing data for that event source row intact.<br><br>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`.<br><br>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.<br><br>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.<br><br>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.<br><br>Default value: 10 (Max: 100.000).<br><br>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.<br><br>Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached. |

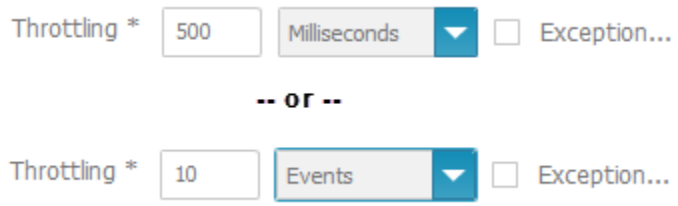
| Property | Required | Description |
|---|---|---|
| | | This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached.<br><br>Set this option to change the bucket strategy to FIFO (first-in, first-out) which discards events for older series (buckets) and stores the newer event in a new series (bucket).<br><br>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.<br><br>Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value.<br><br>Default value: 10;<br><br>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`.<br><br>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`.<br><br>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`.<br><br>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.<br><br>Select one or more attributes that uniquely identify events for this Apama scenario. If multiple fields are required, the order in which you select attributes |

| Property | Required | Description |
|---|---|---|
| | | determines how fields are combined to determine event IDs. |
| Capacity | yes | Enter the maximum number of events to store in this event source. (Max: 100.000)<br>Default value: 10 |
| Memory model | | Determines where events are stored:<br>■ `Internal`: the default which stores events in local memory for this event source.<br>■ `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:<br>■ Change the number of milliseconds to control throttling.<br>■ 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.<br>See "example below." on page 178 |
| Exception | | Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1):<br>■ A number<br>■ `Milliseconds` or `Events` as the measurement for the exception criteria<br>See "example below." on page 178 |

#### Simple and Hybrid Throttling Strategies

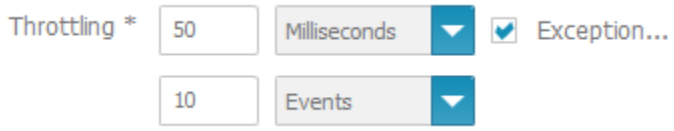
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events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

The default behavior is to push events to views based on a time interval of every 500 milliseconds. You can change the time interval or change the criteria to push events once a minimum count of events are received, such as 10 events. For example:




Simple strategies may still not even out event flow adequately. Instead, you can create hybrid strategies, such as "generally push every 50 milliseconds, but at most 10 events."

Hybrid strategies define the general throttling with the **Throttling** fields. You set the **Exception** option and define the exception that should break the general rule in the **Exception** criteria fields:




With the example hybrid throttling strategy shown above:

- The event source would wait 50 milliseconds after pushing events to subscribing views.
- If less than 10 events are received in that 50 milliseconds, they are pushed at the end of the interval.
- If a tenth event is received within the 50 milliseconds, these 10 events are pushed to subscribing views and both the time interval and the count of events begins again.
- If no events are received within the time interval, the event source waits until it receives an event. When an event is received, the event source pushes this event to subscribing views and restarts the time interval.

On the Apama Event Source overview page you can click on the **Alias** to show a preview of the specific Event Source properties.

## Edit Apama Event Sources

You can edit an already existing Apama Event Source.

**To edite an Apama Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.

3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the **Edit** icon to configure an Apama Event Source.
6. Set the properties for this event source. See table "Apama Event Source properties" on page 180 below.
7. 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 Presto Server starts. Clear this option if you need to manually control startup for this event source. |
| **Apama** URL | yes | URL to the running **Apama** system ( local or remote) |
| **Apama** Scenario | yes | Click **Refresh** to update the list of **Apama** scenarios for the selected Apama Event Source. If the **Apama** URL is set to a valid **Apama** system, then it is possible to select a scenario this event source should subscribe to. |
| Strategy | yes | The strategy that this event source uses for saving and removing events published from the Event Bus. Valid strategies are:<br>■ `BUFFER` = FIFO (first in-first out). Events are stored until event source memory reaches capacity and then the event source removes the oldest events.<br>■ `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.<br><br>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.<br><br>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.<br><br>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.<br><br>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.<br><br>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.<br><br>Default value: 10 (Max: 100.000).<br><br>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. |

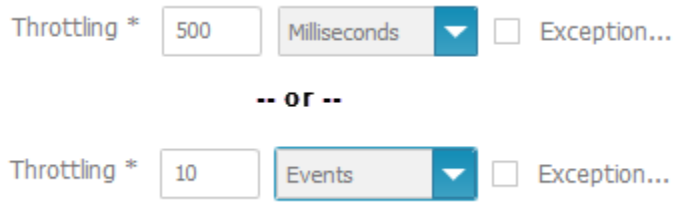
| Property | Required | Description |
|---|---|---|
| | | Determines how additional events are handled if they have new unique values for the dimension that defines buckets in this event source but the maximum number of unique values (buckets) has already been reached.<br><br>This option is clear by default which discards new events with new unique dimension values once the maximum number of buckets has been reached.<br><br>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).<br><br>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.<br><br>Enter the maximum number of events that can be stored in a specific event series (bucket) for each unique dimension value.<br><br>Default value: 10;<br><br>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`.<br><br>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`.<br><br>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`.<br><br>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.<br><br>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)<br><br>Default value: 10 |
| Memory model | | Determines where events are stored:<br><br>■ `Internal`: the default which stores events in local memory for this event source.<br><br>■ `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:<br><br>■ Change the number of milliseconds to control throttling.<br><br>■ 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.<br><br>See "example below." on page 178 |
| Exception | | Set this option to support a hybrid throttling strategy, typically involving both time and event limitations. Then set the exception criteria (Default=1):<br><br>■ A number<br><br>■ `Milliseconds` or `Events` as the measurement for the exception criteria<br><br>See "example below." on page 178 |

### Simple and Hybrid Throttling Strategies

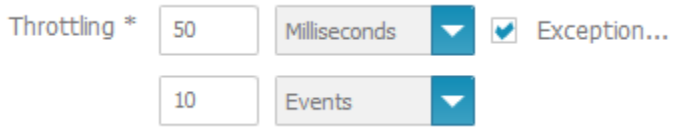
Simple throttling strategies cause an event source to wait for either a specific time interval or for the receipt of a specific number of events and then push all new events to any subscribing real-time views. Throttling can slow event updates to real-time views when the volume or frequency for events 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. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the **Copy** icon to duplicate a specific Apama Event Source.

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

## Delete Apama Event Sources

You can delete an Apama Event Source.

**Note:** Deleting an Apama Event Source may cause data feeds to fail.

**To delete an Apama Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the **Delete** icon to delete a specific Apama Event Source.
6. Click **Save**.

The selected Apama Event Source is deleted from the list.

## Import Apama Event Sources

You can import individual configurations of Apama Event Sources. You can import only Apama Event Sources that are stored in the importexport directory presto-install \mashzone\data\importexport or in one of the subfolders.

The files are saved as archive files, *.mzp. If an Apama Event Source with the same name already exists it cannot be imported. The existing Event Source must be deleted first.

**To import an Apama Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.

4. Open the **Apama** tab.
5. Click **Import Apama Event Source**.
6. Select an **Apama** .mzp file from the list:.
7. Click **Import Apama Event Source**.

The *Apama Event Source* with all relevant settings is imported and displayed.

## Export Apama Event Sources

You can export the configurations of Apama Event Sources. The exported Apama Event Sources are stored directly in the importexport directory presto-install\mashzone\data \importexport. The Apama Event Sources are saved with the corresponding settings and shared as *.mzp archive file.

**To export an Apama Event Source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the **Apama** tab.
5. Click the Export button to export an **Apama** Event Source.
6. Click **Close**.

The selected **Apama** Event Source is saved in the importexport directory as a mzp archive file.

# Start or Stop an Event Source

To begin receiving events from the Event Bus, you must start the event source configured for that event type. You can also stop individual event sources.

**Note:** Stopping an event source causes any existing events currently stored in memory to be deleted.

**To stop/start an event source:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Open the event source tab **EDA** or **Apama** and either:
   - Select a specific event source and click **Start** to start just that event source.

- Or select a specific event source and click **Stop** to stop that event source.

The selected event sources are stopped respectively started.

## Restart all Event Source

To begin receiving events from the Event Bus, you must start the event source configured for that event type. You can restart all event sources at once.

> **Note:** Restarting all event sources causes any existing events currently stored in memory to be deleted.

**To restart all event sources:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Event Service** to expand this section of the Administration menu.
3. Click **Event Service**. The **Event Service** page will be displayed.
4. Click **Restart all** to restart all event source instances of the selected event source type.

All event sources are restarted.

# 7 Process Performance Manager (PPM) Integration

Software AG Process Performance Manager (PPM) lets you discover and analyze processes that are not formally managed by a business process management solution (*BPMS*), such as webMethods BPMS. Using data sources throughout your enterprise, such as transactional data from your business systems, event streams from webMethods BPMS or database records from trading partners, PPM can model a process and assess its performance across various dimensions, such as region, product line, volume, or time. You can also use PPM's analytic tools to mine other data in your enterprise for meaningful patterns, trends, or correlations.

Information from PPM can be used in two ways in Presto:

- As a source of data for a MashZone feed.
- As charts to add to a workspace app in Mashboard using the PPM Chart app.

**Note:** Presto is compatible with PPM version 5.1.0 or above.

Connections to PPM are used in both cases. The tasks to integrate Presto with PPM are:

1. Create PPM Connections for use in data sources or charts.

   You can also Manage PPM Connections.

2. Import the PPM Chart App to allow users to select and add PPM charts to workspace apps.

# Manage PPM Connections

**To edit, delete, import and export PPM connections**

1. Click Admin Console in the Presto Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**.
4. Follow the procedure of the remaining steps:
   - "Edit PPM Connections" on page 192
   - "Delete PPM Connections" on page 193
   - "Export PPM Connections" on page 194
   - "Import PPM Connections" on page 193

# 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 Presto.

**Note:** Presto is compatible with PPM 5.1.0 or above.

For Presto 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 Presto determine them using the URL of a PPM favorite (favorites path). For information on copying the URL of a PPM favorite, see PPM on-line help topics.

**To create PPM connections**

1. Click  Admin Console in the Presto Hub main menu.
2. Click **Connections** to expand this section of the Administration menu.
3. Click **PPM connections**.
4. Click **Create**.
5. Enter a name for the **PPM connections** in the **Alias** field, for example, the client name. The connection data is saved under this alias. Users may choose **PPM connections** by their alias.
6. To retrieve the connection data from the URL of a favorite from PPM:
   a. Click **Retrieve data**.
   a. Enter the URL of the PPM favorite that you copied earlier in the **URL** field.
   a. Click **Resolve URL** to retrieve the required parameters from the URL. Presto uses the favorite URL to complete the remaining fields for this connection.
7. To enter connection information manually:
   a. Select the protocol (HTTP or HTTPS) to use for the web application server that hosts the PPM query interface.
   a. In the **Host** field, enter the domain name or IP address of the PPM load balancer.
   a. In the **Port** field, enter the port number of the PPM load balancer.
   a. Specify the remaining context of the PPM query interface in the **Context** field. The context has the format **ppm/API_client name**, such as, API_umg_en.
8. Click **Check availability** to verify that the data is correct and that the PPM client is available.
9. Click **Save**.

The PPM connection is created and listed by alias name. This also lists the PPM version and availability of the PPM client.

# Edit PPM Connections

You can edit already existing PPM connections.

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

For Presto 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 Presto determine them using the URL of a PPM favorite (favorites path). For information on copying the URL of a PPM favorite, see PPM on-line help topics.

### To edit PPM connections

1. Click  Admin Console in the Presto Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click the **Edit** icon to configure a PPM connection.
5. The **Alias** field of an already configured **PPM connection** is not editable. The connection data is saved under this alias.
6. To retrieve the connection data from the URL of a favorite from PPM:
   a. Click **Retrieve data**.
   a. Enter the URL of the PPM favorite that you copied earlier in the **URL** field.
   a. Click **Resolve URL** to retrieve the required parameters from the URL. Presto uses the favorite URL to complete the remaining fields for this connection.
7. To enter connection information manually:
   a. Select the protocol (HTTP or HTTPS) to use for the web application server that hosts the PPM query interface.
   a. In the **Host** field, enter the domain name or IP address of the PPM load balancer.
   a. In the **Port** field, enter the port number of the PPM load balancer.
   a. Specify the remaining context of the PPM query interface in the **Context** field. The context has the format **ppm/API_client name**, such as, API_umg_en.

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

Your changes are applied.

## Delete PPM Connections

You can delete existing PPM connections.

**Note:** Deleting PPM connections may cause data feeds or PPM Chart app to fail.

**To delete PPM connections:**

1. Click Admin Console in the Presto Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click the **Delete** icon to delete a PPM connection.
5. Confirm the deletion.

The selected PPM connections are deleted from the list.

## Import PPM Connections

You can import individual PPM connections.

The imported PPM connections are stored in the presto-install/mashzone/data/importexport folder with the corresponding settings and shares as MashZone *.mzp archive files. The export archive files have names in the form *A_PPM_PPM connection name_time stamp of export.mzp*.

**To import PPM connections**

1. Click Admin Console in the Presto Hub main menu.
2. Click **Connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click the **Import**.
5. Click **Find** and select a PPM connection file.

The selected PPM connection with all relevant settings is imported and displayed.

# Export PPM Connections

You can export individual PPM connections.

The exported PPM connections are stored in the presto-install/mashzone/data/importexport folder with the corresponding settings and shares as MashZone *.mzp archive files. The export archive files have names in the form *A_PPM_PPM connection name_time stamp of export.mzp.*

### To export PPM connections

1. Click  Admin Console in the Presto Hub main menu.
2. Click **PPM connections** to expand this section of the Administration menu.
3. Click **PPM connections**. A list of all available PPM connections will be displayed.
4. Click **Export** to export a PPM connection.
5. Click **OK**.

The selected PPM connection is saved in the importexport directory as an archive file.

# Import the PPM Chart App

The PPM Chart app, allows users to add charts to a dashboard for business process performance that were created in PPM. Users add PPM charts to workspace apps (dashboards) in Mashboard.

This custom app is available as an import package that is included in the Presto installation. If integration with PPM is needed, you can simply import this custom app to make it available to users.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Copy the *presto-install* /mashzone/clientapps/ppm-chart.zip file to the *presto-install* /prestocli/bin folder.
3. Open a command window and move to the *presto-install* /prestocli/bin folder.
4. Enter this command:

   `padmin importApps -f ppm-chart.zip -s -u username-w password`

   With the appropriate account credentials. This typically in an administrator account.

   The `-s` flag ensures that this also imports the mashable or mashup dependencies for this custom app. There are other flags you may use. You may also specify a remote Presto Server. See "Importing App Metadata" on page 247 for more information and examples on this command.

Messages and errors from the import process are sent to the console window (stdout).

5. Once the import is successfully finished, find and open this app in Presto Hub.
6. To allow other users to work with this app, add run permissions for appropriate users and groups.

# 8 Presto and MashZone Repositories

The Presto Repository is the database that the Presto Server uses to store meta-data, attributes and configuration for Presto including:

- Artifacts (mashable information sources, mashups, and apps)
- Macros
- Taxonomies such as categories, tags and providers
- Presto attributes (global, user and custom for artifacts)
- Configuration properties for the Presto 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 Presto Repository.

**Important:** The Presto Repository and MashZone Repository are initially installed in a Derby database suitable *only for trial*purposes. For proof-of-concept, development or production uses, move both these repositories to a robust and compatible solution.

Similarly, the MashZone Repository holds meta-data for the MashZone feeds and connections supported by the Integrated MashZone Server for Presto.

Configuration and administration tasks for these two repositories include:

- Move the Presto and MashZone Repositories to a Robust Database Solution
- Support International Character Sets and Locales
- Use the Default Presto User Repository
- Change Presto Repository Ports
- "Tuning the Presto or MashZone Repository Connection Pool" on page 199
- Synchronize the Presto Repository and Presto Server Time Zones
- Sharing the Presto Repository in Clustered Environments
- Configure BigMemory Servers for Presto Caching and In-Memory Stores
- Maintenance Suggestions

#### Maintenance Suggestions

Your existing standards for database backups, security and maintenance can be applied to the Presto and MashZone Repositories. In addition, you should set up procedures to monitor or regularly manage growth for the Presto Auditable Events table. This table tracks audit information for updates to the Presto 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 Presto or MashZone Repository Connection Pool

In addition to basic connection configuration, you can configure the connection pools for the Presto Repository or the MashZone Repository. In many cases, you need to tune this configuration to optimize your Presto environments.

**Note:** For a complete list of connection properties, see TomEE Datasource Properties.

To tune the connection pool, you update properties in the `<Resource>` element for the Presto or MashZone repository in the *presto-install* /apache-tomee-jaxrs/conf/tomee.xml file and then restart Presto 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.<br>If this is enabled, idle connections are tested using the `Validation query`. See "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 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 Presto Repository and Presto Server Time Zones

Creation and modification timestamps for artifacts and other Presto Repository metadata can be different than times when events occurred in the Presto Server in two cases:

- If the server hosting the Presto Repository is located in a different time zone from the server hosting the Presto Server
- If the time zone setting for the database hosting the Presto Repository is set to a different time zone from the server hosting the Presto Server

You can correct this problem by specifying a time zone in configuration for the Presto 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 *presto-config* folder. This is either in a shared external configuration folder or in *app-server:port* /presto/WEB-INF/classes.
2. Update the following properties:
   - `jdbc.url=jbdc:mysql://`*`hostname/database-name`*`?useLegacyDatetimeCode=false`
   - `dbServer.timeZone=`*`time-zone-id-for-mysql-server`*

     Valid time zone identifiers include GMT time zones, common abbreviations such as EST or UTC, or time zone locations such as `America/Los_Angeles`.
3. Save these changes.

4. Restart the Presto Server. See "Start and Stop the Presto Server" on page 20 for instructions.

# 9 Presto Server Administration

Basic administration tasks for the Presto Server include:

- Start and Stop the Presto Server

  See also Startup Considerations.
- View Presto Logs and Purge the Audit Log
- Manage Files for Presto Features or Artifacts
- Upgrade to New Versions of Presto and Migrate Artifacts
- Deploying Presto Instances, Clusters or Artifacts
- Clustering Presto Servers

# View Presto Logs

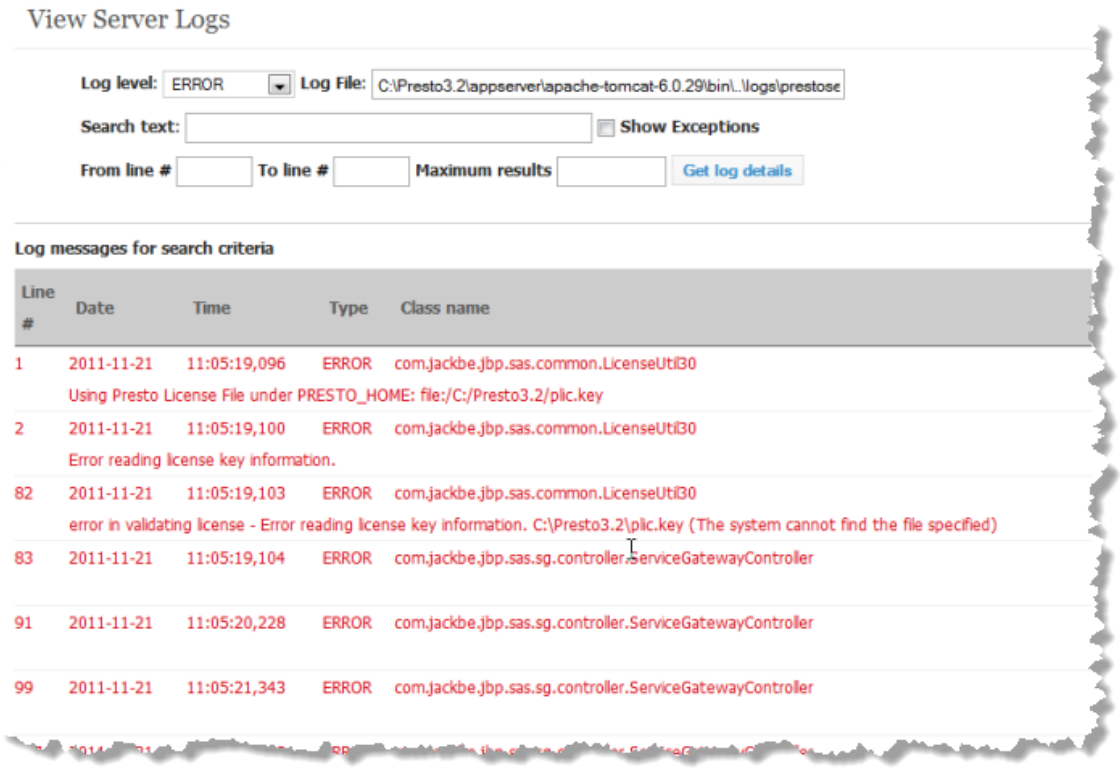
You can view two Presto logs in Presto Hub. See "View the Presto Server Log" on page 204 and "View the Audit Log for a Mashable, Mashup or App" on page 205 for instructions. See also "Presto Logging" on page 103 for links to additional logging options for Presto.

## View the Presto Server Log

To view the Presto Server log, Click  Admin Console in the Presto 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 Presto 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.

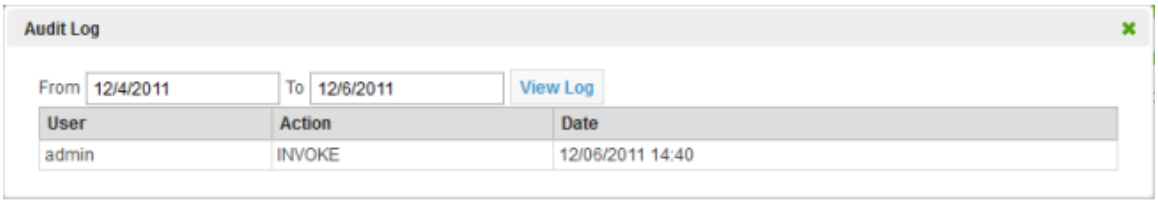
## View the Audit Log for a Mashable, Mashup or App

The Audit Log can track each invocation or load for mashables, mashups or apps in Presto 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 105).

**Note:** Purging the Audit Log permanently deletes audit data. You may wish to make a backup of this data in the Presto Repository before completing this purge.

1. Click Admin Console in the Presto 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 Presto. See "Manage Pluggable Libraries" on page 206, "Manage Pluggable Views" on page 207 and "Manually Changing the Default Version for Libraries" on page 208 for more information.

## Manage Pluggable Libraries

Pluggable libraries and pluggable view libraries are added to Presto 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 Presto configuration is up to date.

### To manage pluggable libraries

1. Click Admin Console in the Presto 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 Presto 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 208 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 Presto View Gallery so that users may add this view to any number of mashables and mashups. They are added to Presto 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 Presto configuration is up to date.

**To manage pluggable libraries**

1. Click Admin Console in the Presto 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 Presto 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 208 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 Presto, 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 Presto 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 Presto Features or Artifacts

Presto 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. Presto 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 Presto Repository to ensure better management of resources and easier deployment or migration across different environments and versions.

Most files are added to the Presto Repository automatically when users register, create or update the corresponding mashables, apps and macros. Presto administrators may also need to manually add files to Presto 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 Presto.

Common management tasks for files include: Add External Resources as Presto Files, Find Presto Files or Update or Delete Presto Files. See also "File Organization" on page 210 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 206 .

## Add External Resources as Presto Files

You can add external resources to Presto 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 Presto**

1. Click Admin Console in the Presto Hub main menu.
2. Expand the Platform Features section and click **File Resources**.
3. Click **Upload New Files**.
4. Click **Browse**, find and select the file you want to upload and click **Open**.

   The location and file name fill in and a new set of fields open to upload another file.
5. If needed, add to the path or change the file name.

   The name of the file defaults to */file-name* . If you accept the default, the URL to access this file becomes `http://`*app-server:port* `/presto/files/`*file-name* .

   You can organize files into 'pseudo folders' by adding to the path, using `/` as the separator. For example, a file name of `/images/reports.png` has a URL of `http://`*app-server:port* `/presto/files/images/reports.png` and can be found in file search (along with any other files in the 'images folder') by searching for `images` as the file name.

   You can also upload files that are normally loaded automatically, such as thumbnails. Simply specify the standard path. See "File Organization" on page 210 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 Presto Repository and are now available via a Presto URL.

## Find Presto Files

**To find files that have been uploaded to Presto**

1. Click Admin Console in the Presto 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 the file(s). See "File Organization" on page 210 for a list of the standard paths that Presto uses.

4. Click **Search**.

> **Note:** File search results are always sorted by path and file name.

## Update or Delete Presto Files

Although rare, you may occasionally need to update or even delete files from Presto.

**To update or delete a file**

1. Click  Admin Console in the Presto 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 Presto Files" on page 209 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 Presto.
   c. Click **Upload this file**.
5. To delete a file, click **Delete** on the line for that file.

## File Organization

File names are path-like to support both URL access and common file organization techniques. Files that are uploaded automatically, such as resources for custom apps or views, are organized in the `/system` 'folder.' Files that you upload manually default to the 'root folder' of `/`. You can, however, define any level of folder organization you need by defining your own folder paths when you manually upload files.

Standard file paths include:

- `/system/lib`: the foot path for all pluggable views.
- `/system/mashlets`: the root path for all custom apps.
- `/system/mashlets/`*`app-id`*: the root path for all resource files for a specific custom app.
- `/system/mashlets/`*`app-id`*`/js`: JavaScript libraries for a specific custom app
- `/system/mashlets/`*`app-id`*`/css`: CSS stylesheets for a specific custom app.
- `/system/mashlets/`*`app-id`*`/screenshots`: screenshots for a specific custom app.

- `/system/thumbnails`: root folder for thumbnails.
- `/system/wires/schemas`: root folder for all registered XML schemas available for use in the Mapper block.

To determine the URL to access a file, add `http://`*app-server:port* `/presto/files` to the file name.

# Upgrade to New Versions of Presto and Migrate Artifacts

Effective with version 3.8, documentation for upgrades and migration of artifacts has moved:

- *PrestoCore*: see the *Upgrading Software AG Products* Guide.
- *Presto Add-On for Portals*
- *Presto Add-On for SharePoint*: see the upgrade section in the Installation and Configuration Guide for Presto Add-On for SharePoint for SharePoint 2007 or 2010.

Migrating artifacts generally involves export and import commands. See "Deploying Presto Artifacts and Other Metadata" on page 232 for instructions.

## Migrating Extensions to Presto

Presto includes extension points in many places where you can create custom functionality. Typically, these extensions involve custom source code or configuration or both. Extensions are typically deployed in an external configuration folder or in the Presto Server web application.

Migrating these extensions generally involves copying over any custom classes, scripts or other source files and updating configuration in the new Presto version.

### Migrating Custom XPath Functions, XSLT Stylesheets, Java Classes or JRuby Scripts for Mashups

You can define custom XPath functions and add them to the Presto Server to perform specific formatting or calculations in mashups. These XPath functions are implemented as Java classes that are deployed as class or JAR files in the external configuration folder or in the Presto Server.

You can also use mashups to wrap POJO services or use Java classes or JRuby scripts within mashups to handle custom logic. The code for these extensions is also deployed in the external configuration folder or in the Presto Server.

You can also call XSLT stylesheets to perform transformations on mashup data using the <xslt> statement. These stylesheets are also deployed in theexternal configuration folder or in the Presto Server.

All of these mashup extensions must be deployed in your migration target:

- If you have deployed these extensions in an external configuration folder, you can simply add this folder to the classpath for the application server hosting your migration target or create an external configuration folder for the target and copy these extensions. See "Setting Up an External Presto Configuration Folder" on page 258 for more information.
- If you deployed these extensions in either:
  - *source-web-apps-home* /presto/WEB-INF/classes
  - *source-web-apps-home* /presto/WEB-INF/lib

  Simply copy them to the target external configuration folder or to these folders in the target Presto Server, respectively:
  - *target-web-apps-home* /presto/WEB-INF/classes
  - *target-web-apps-home* /presto/WEB-INF/lib

If you use JRuby scripts, you must also install and configure JRuby for the migration target.

## Migrating Macro Libraries and Custom Macros for Mashups

Macros are user-defined statements that you can use in mashups. Macros can be shared, and thus accessible to many mashups, in macro libraries or in the built-in global macro library for the Presto Server. In version 2.7.0 and earlier, these macro libraries were either:

- *.emml-macros files deployed in *existing-web-apps-home* /presto/WEB-INF/classes
- JAR files deployed in *existing-web-apps-home* /presto/WEB-INF/lib

With version 3.0, macros are registered in the Presto Repository. You can also register your custom macros in *domains* to keep macros organized. This is very useful for macros used as custom action blocks in Wires as each macro domain appears as a separate category of action blocks.

### To migrate macros to the target

1. If you want to use macro domains to organize your existing macros:
   a. Move any custom macros from global.emml-macros into separate macro library files for each domain you want to create.
   b. Update the namespace of your custom macro libraries to the EMML namespace for this release.
   c. Add domain names to your custom macro libraries.
   d. Save your changes.
2. Register your custom macros using the mashup command utilities.

## Migrating the Presto Repository from 3.0 to 3.1

Migration of artifacts or data is not required when upgrading Presto from version 3.0 to version 3.1. There have, however, been minor additions or updates to configuration properties that must be added to or updated in a 3.0 Presto Repository to use that repository for Presto 3.1.

To add or update these properties, you must use the database management tool for the database hosting the Presto Repository to run a set of SQL statements. For HSQL, the default database:

1. Open a command window and move to the *presto-install* /prestorepository/hsqldb folder.
2. Run the manager.bat script for Windows environments or the manager.sh script for Linux, OS X or UNIX environments and log in using an account with privileges to insert records.
3. Enter and run these SQL commands:

```
UPDATE CONFIGURATION SET CONFIGVALUE = 'false' WHERE CONFIGKEY = 'security.filtersearchresult
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('security.showOnlyExecutableItems',
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('snapshot.on', 'false');
UPDATE CONFIGURATION SET CONFIGVALUE = 'serviceURL, Etag, Expires, Last-modified, Cache-Contro
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('http.socket.timeout','0');
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('http.connection.timeout','0');
```

   You may also need to add a commit statement to finalize the update.

## Migrating the Presto Repository from 3.1.1 to 3.2

If you are upgrading to Presto 3.2, but using an existing 3.1.1 Presto Repository, there are several minor updates to the schema or configuration properties that you must make to your 3.1.1 Presto Repository.

1. If your existing 3.1.1 Presto Repository is hosted in a PostgreSQL database, you *must* replace one JAR file before you apply schema changes:
   a. Remove the *web-apps-home* /presto/WEB-INF/lib/rds.jar JAR file. You can delete this JAR or simply move it to a folder outside of the Presto web application.
   b. Copy this JAR file:

      *presto-install* /dist/rds-postgresql.jar

      To the *web-apps-home* /presto/WEB-INF/lib folder.
2. Use the SQL client or command line tool for your existing 3.1.1 Presto Repository. Run the SQL script shown below from the folder that corresponds to your database in *presto-install* /prestorepository:

| Database | Folder | SQL Script |
|---|---|---|
| HSQL | hsqldb | updateDB3.2.sql |
| Microsoft SQL Server | mssqldb | |
| MySQL | mysqldb | |
| Oracle | oracledb | |
| PostgreSQL | postgresdb | |

## Migrate the Presto Repository from 3.2.1 to 3.5

If you are upgrading to Presto 3.5, but using an existing 3.2.1 Presto Repository, there are a few minor updates to the schema or configuration properties that you must make to your 3.2.1 Presto Repository.

Use the SQL client or command line tool for your existing 3.2.1 Presto Repository and execute the folowing script based on the database host for Presto Repository:

- Presto 3.5 Schema Updates for MS SQL Databases
- Presto 3.5 Schema Updates for MySQL Databases
- Presto 3.5 Schema Updates for Oracle Databases
- Presto 3.5 Schema Updates for PostGres Databases

If your Presto Repository is hosted in a Microsoft SQL Server, MySQL or Oracle database, you must also replace the JAR file for the Presto Server. See "Upgrade the Presto 3.5 Repository JAR for MSSQL, MySQL or Oracle" on page 216 for instructions.

### Presto 3.5 Schema Updates for MS SQL Databases

```
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE THE FILE IS GONE
DELETE RO_FILE_ASSOC
FROM RO_FILE_ASSOC
LEFT JOIN "FILE"
  ON
   (
     RO_FILE_ASSOC.FILE_ID = "FILE".ID
   )
WHERE "FILE".ID IS NULL;
-- ALTER RO_ID TO MATCH REGISTRYOBJECT.ID
ALTER TABLE RO_FILE_ASSOC ALTER COLUMN RO_ID VARCHAR(700);
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE REGISTRY OBJECT IS GONE
DELETE RO_FILE_ASSOC
FROM RO_FILE_ASSOC
LEFT JOIN REGISTRYOBJECT
```

```
  ON
    (
      RO_FILE_ASSOC.RO_ID = REGISTRYOBJECT.ID
    )
WHERE REGISTRYOBJECT.ID IS NULL;
-- ADD THE CONSTRAINTS
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_FILE_FK1 FOREIGN KEY (FILE_ID) REFERENCES "FILE"
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_RO_FK2 FOREIGN KEY (RO_ID) REFERENCES REGISTRYOB
-- ALTER EXISTING VALUE
UPDATE CONFIGURATION SET CONFIGVALUE='<html><body>Your password has been updated.  Your new passw
If you have any questions about this change, please contact your administrator.</body></html>'
WHERE CONFIGKEY = 'email.template.body.passwordchanged';
-- INSERT NEW VALUE
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('auditlog.on.excludeActions', 'USER_LO
-- FORCE RELOAD SECURITY PROFILES ON NEXT RESTART TO PICKUP NEW PROFILES
UPDATE INIT_TRACKING SET REQUIRE_INIT=TRUE WHERE COMPONENT_NAME = 'SecurityProfileDefinitions'
```

## Presto 3.5 Schema Updates for MySQL Databases

```
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE THE FILE IS GONE
DELETE RO_FILE_ASSOC
FROM RO_FILE_ASSOC
LEFT JOIN FILE
  ON
    (
      RO_FILE_ASSOC.FILE_ID = FILE.ID
    )
WHERE FILE.ID IS NULL;
-- ALTER RO_ID TO MATCH REGISTRYOBJECT.ID
ALTER TABLE RO_FILE_ASSOC MODIFY COLUMN RO_ID VARCHAR(700);
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE REGISTRY OBJECT IS GONE
DELETE RO_FILE_ASSOC
FROM RO_FILE_ASSOC
LEFT JOIN REGISTRYOBJECT
  ON
    (
      RO_FILE_ASSOC.RO_ID = REGISTRYOBJECT.ID
    )
WHERE REGISTRYOBJECT.ID IS NULL;
-- ADD THE CONSTRAINTS
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_FILE_FK1 FOREIGN KEY (FILE_ID) REFERENCES FILE (
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_RO_FK2 FOREIGN KEY (RO_ID) REFERENCES REGISTRYOB
-- ALTER EXISTING VALUE
UPDATE CONFIGURATION SET CONFIGVALUE='<html><body>Your password has been updated.  Your new passw
If you have any questions about this change, please contact your administrator.</body></html>'
WHERE CONFIGKEY = 'email.template.body.passwordchanged';
-- INSERT NEW VALUE
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('auditlog.on.excludeActions', 'USER_LO
-- FORCE RELOAD SECURITY PROFILES ON NEXT RESTART TO PICKUP NEW PROFILES
UPDATE INIT_TRACKING SET REQUIRE_INIT=TRUE WHERE COMPONENT_NAME = 'SecurityProfileDefinitions'
```

## Presto 3.5 Schema Updates for Oracle Databases

```
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE THE FILE IS GONE
DELETE FROM RO_FILE_ASSOC
WHERE RO_FILE_ASSOC.ID IN(
  SELECT RO_FILE_ASSOC.ID
  FROM RO_FILE_ASSOC LEFT OUTER JOIN "FILE" ON "FILE".ID=RO_FILE_ASSOC.FILE_ID
  WHERE "FILE".ID IS NULL);
-- ALTER RO_ID TO MATCH REGISTRYOBJECT.ID
ALTER TABLE RO_FILE_ASSOC MODIFY RO_ID VARCHAR2(700);
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE REGISTRY OBJECT IS GONE
```

```
DELETE FROM RO_FILE_ASSOC
WHERE RO_FILE_ASSOC.ID IN (
  SELECT RO_FILE_ASSOC.ID
  FROM RO_FILE_ASSOC LEFT OUTER JOIN REGISTRYOBJECT ON RO_FILE_ASSOC.RO_ID = REGISTRYOBJECT.ID
  WHERE REGISTRYOBJECT.ID IS NULL
  );
-- ADD THE CONSTRAINTS
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_FILE_FK1 FOREIGN KEY (FILE_ID) REFERENCES "FILE"
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_RO_FK2 FOREIGN KEY (RO_ID) REFERENCES REGISTRYOB
-- ALTER EXISTING VALUE
UPDATE CONFIGURATION SET CONFIGVALUE='<html><body>Your password has been updated.  Your new passw
If you have any questions about this change, please contact your administrator.</body></html>'
WHERE CONFIGKEY = 'email.template.body.passwordchanged';
-- INSERT NEW VALUE
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('auditlog.on.excludeActions', 'USER_LO
-- FORCE RELOAD SECURITY PROFILES ON NEXT RESTART TO PICKUP NEW PROFILES
UPDATE INIT_TRACKING SET REQUIRE_INIT=TRUE WHERE COMPONENT_NAME = 'SecurityProfileDefinitions'
```

### Presto 3.5 Schema Updates for PostGres Databases

```
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE THE FILE IS GONE
DELETE FROM RO_FILE_ASSOC
WHERE RO_FILE_ASSOC.ID IN (
  SELECT RO_FILE_ASSOC.ID
  FROM RO_FILE_ASSOC
  LEFT JOIN FILE ON FILE.ID = RO_FILE_ASSOC.FILE_ID
  WHERE FILE.ID IS NULL);
-- ALTER RO_ID TO MATCH REGISTRYOBJECT.ID
ALTER TABLE RO_FILE_ASSOC ALTER COLUMN RO_ID TYPE VARCHAR(700);
-- REMOVE DANGLING RECORDS FROM RO_FILE_ASSOC WHERE REGISTRY OBJECT IS GONE
DELETE FROM RO_FILE_ASSOC
WHERE RO_FILE_ASSOC.ID IN (
  SELECT RO_FILE_ASSOC.ID
  FROM RO_FILE_ASSOC
  LEFT JOIN REGISTRYOBJECT
    ON
     (
          RO_FILE_ASSOC.RO_ID = REGISTRYOBJECT.ID
     )
  WHERE REGISTRYOBJECT.ID IS NULL);
-- ADD THE CONSTRAINTS
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_FILE_FK1 FOREIGN KEY (FILE_ID) REFERENCES FILE (
ALTER TABLE RO_FILE_ASSOC ADD CONSTRAINT RO_FILE_RO_FK2 FOREIGN KEY (RO_ID) REFERENCES REGISTRYOB
-- ALTER EXISTING VALUE
UPDATE CONFIGURATION SET CONFIGVALUE='<html><body>Your password has been updated.  Your new passw
If you have any questions about this change, please contact your administrator.</body></html>'
WHERE CONFIGKEY = 'email.template.body.passwordchanged';
-- INSERT NEW VALUE
INSERT INTO CONFIGURATION (CONFIGKEY, CONFIGVALUE) VALUES ('auditlog.on.excludeActions', 'USER_LO
-- FORCE RELOAD SECURITY PROFILES ON NEXT RESTART TO PICKUP NEW PROFILES
UPDATE INIT_TRACKING SET REQUIRE_INIT=TRUE WHERE COMPONENT_NAME = 'SecurityProfileDefinitions'
```

## Upgrade the Presto 3.5 Repository JAR for MSSQL, MySQL or Oracle

Presto 3.5 ships with two JAR files for the Presto Server to connect to and work with the Presto Repository. By default, it is installed with the JAR that is compatible with the default Presto Repository hosted in a Derby database.

If you are upgrading to version 3.5, but need to use your existing Presto Repository hosted in a Microsoft SQL Server, MySQL or Oracle database, you must switch out these JARs, following these steps:

1. Remove the *web-apps-home* /presto/WEB-INF/lib/rds-postgresql-derby.jar JAR file for each Presto Server that uses this Presto 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 Presto.
2. Copy this JAR file:

   *presto-install* /dist/rds-oracle-mysql-mssql.jar

   To the *web-apps-home* /presto/WEB-INF/lib folder.
3. Restart the Presto Server. For instructions, see "Start and Stop the Presto Server" on page 20.

## Migrating Existing Apps from Presto 3.1.1 to 3.2

Migrating to Presto 3.2 requires minor updates to existing apps from previous versions:

- Apps in version 3.2 have additional metadata in their App Specification that identifies what devices they are compatible with. Metadata tags for desktop compatibility must be added to all existing apps.

  **Note:** If any existing apps are also compatible with mobile phone or tablet devices, you can update metadata tags in their app Specifications manually after you have finished this migration step. .

- The index.html file for existing custom apps includes the PrestoLibrary Loader library. The URL to this library has changed in Presto 3.2 and must be updated to allow these apps to work properly.

Once you have installed Presto 3.2, pointed this release to your existing Presto Repository, and successfully started the Presto Server, you can use an internal Presto API to update this URL in all your custom apps at once:

1. Login to Presto 3.2 as an administrator.
2. In the same browser, enter the following URL, with the appropriate server and port for your environment, to invoke this API:

   `http://`*app-server:port* `/presto/edge/api/rest/MashletHub/upgradeApps`

   The response should be empty (no errors shown).

This updates the appropriate URL in all custom apps and adds desktop compatibility tags to all apps in the Presto Repository for the Presto Server where you are logged in.

## Updating Built-in Policies for Presto 3.2

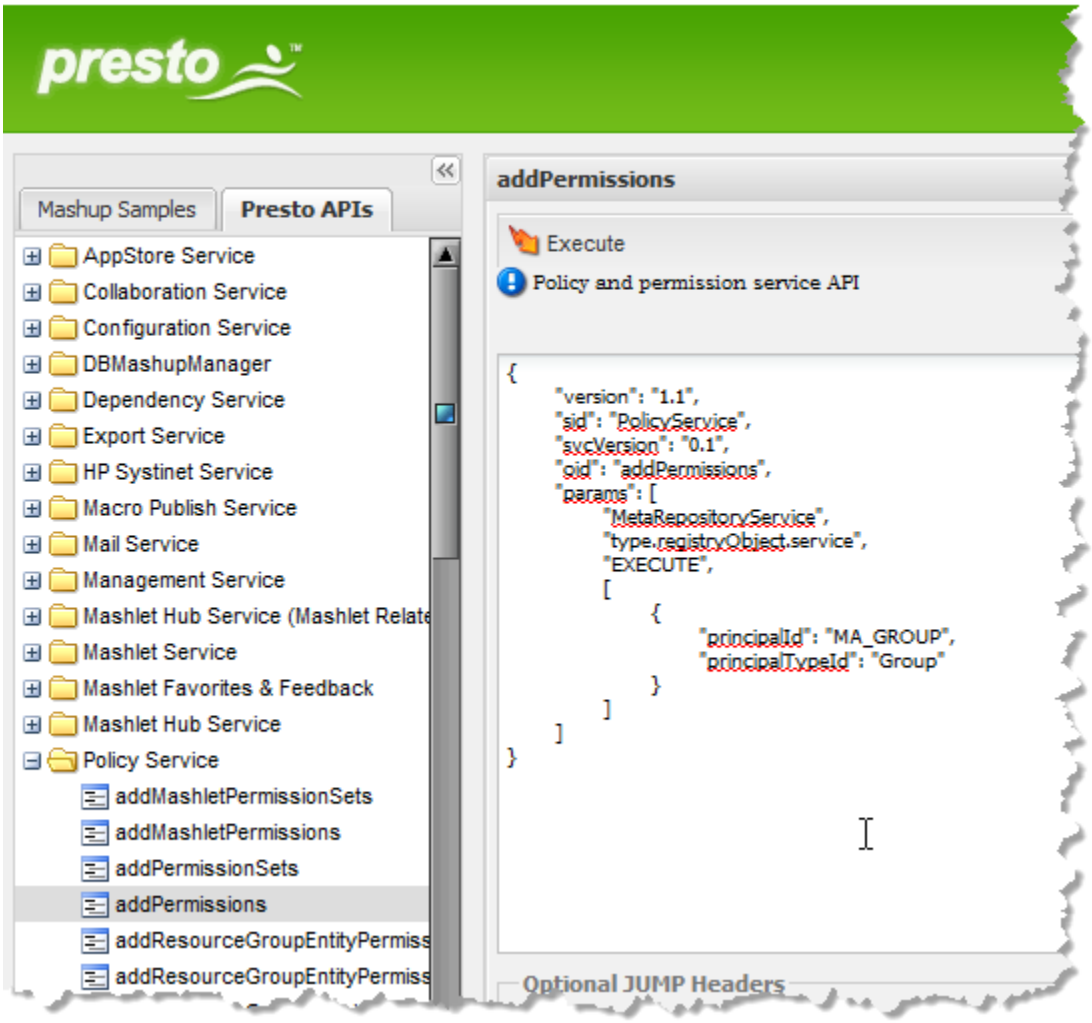
The 3.2 release has added two built-in policies which you must add to your existing Presto Repository when you migrate from 3.1.1 to 3.2.

1. Log in to Presto 3.2 as an administrator.
2. Open a browser page with this URL, substituting the application server and port where Presto 3.2 is hosted:

   `http://`*app-server:port* `/presto/dev`

   **Important:** This is an internal Presto tool that exposes internal APIs. You should not use these APIs unless specifically instructed by Presto Technical Support.

3. Click the **Presto APIs** tab and expand the **Policy Engine** folder.
4. Select the `addPermissions` method. The content pane exposes a sample request for this API in JUMP format:




5. Change the properties in the array for the `params` property to:

```
"params" : [ "MetaRepositoryService.getUserTags",
             "type.registryObject.operation",
             "EXECUTE",
             [ { "principleId":"Presto_Guest",
               "principleTypId":"SpecialGroup" } ]
```

   Then click **Execute**.

6. Select the `addPermissions` method and change the properties in the array for the `params` property to:

```
"params" : [ "MetaRepositoryService.getSystemTags",
             "type.registryObject.operation",
             "EXECUTE",
             [ { "principleId":"Presto_Guest",
               "principleTypId":"SpecialGroup" } ]
```

   Click **Execute**.

7. Select the `addPermissions` method and change the properties in the array for the `params` property to:

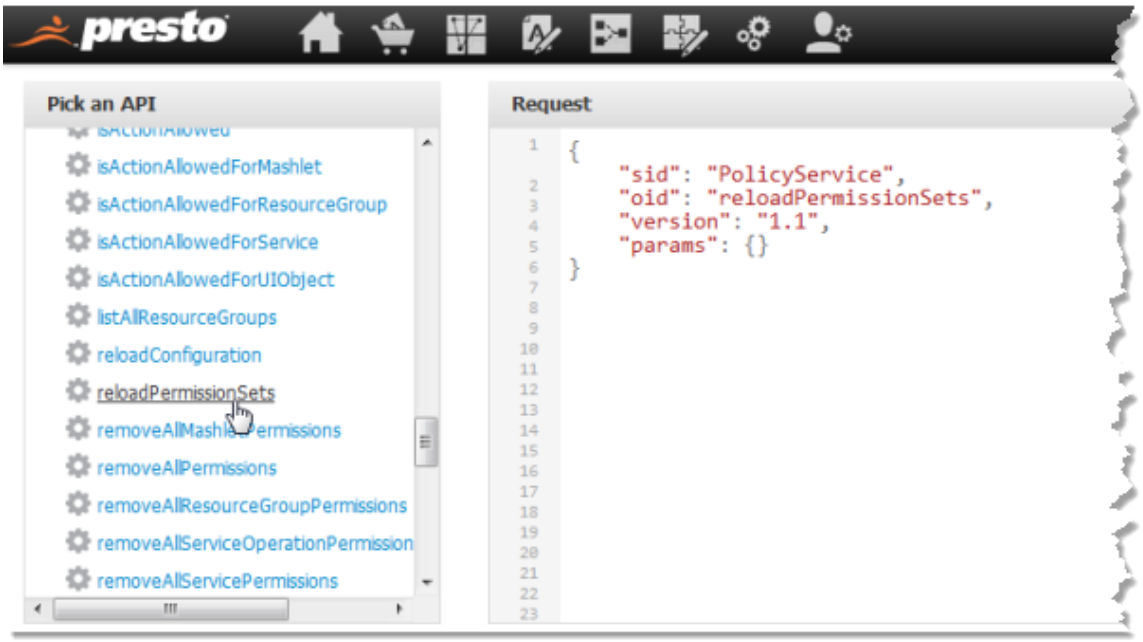
```
"params" : [ "JEMSDesigner.updateMashupScript",
             "type.registryObject.operation",
             "EXECUTE",
             [ { "principleId":"Presto_PowerUser",
               "principleTypId":"SpecialGroup" } ]
```

   Click **Execute**.

## Updating Built-In Policies for Presto 3.5

If you are upgrading to Presto 3.5 but using your existing 3.2.1 Presto Repository, you must add some built-in Presto policies to support new features in this release. You use a platform API in the API Console to update these policies:

1. Log in to Presto 3.5 with an account that has administrator or developer privileges.
2. Click  to open the **API Console**.
3. Find and expand the `Policy Service`.
4. Select the `reloadPermissionSets` operation. The request for this operation opens and should look like this:




5. Click **Run** in the Request panel. You will see a successful reponse such as this:


## Updating Built-In Policies for Presto 3.9

The 3.9 release has added a built-in policy which you must add to your existing Presto Repository when you migrate from 3.5 or higher to 3.9.

1. Login to Presto and click Admin Console in the Presto Hub main menu.
2. Expand the Security & Policies section and click **Reset Policies**.
3. Click **Reset Default Policies**.

This resets the default policies for 3.9 which also adds the new policy Presto requires. This does not affect any custom policies you have already put in place.

## Updating the Macro Metadata Service for Presto 3.2

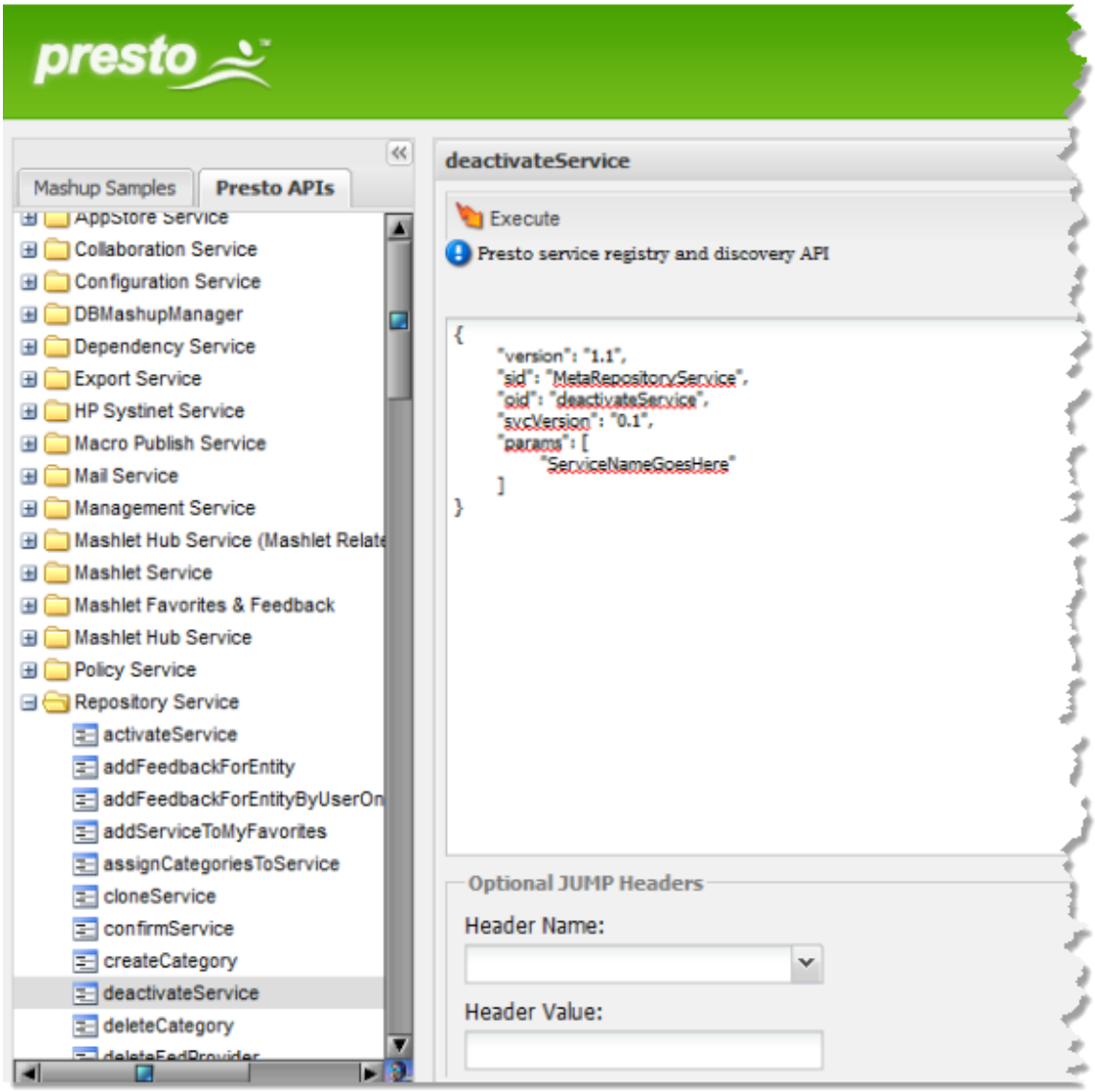
An internal Presto service, the MacroMetadataService, has been enhanced in version 3.2 to support enhancements in Wires for better control of the user interface for custom blocks. To update this service to the 3.2 version, you must delete the existing registered service from your previous Presto version:

1. Log into Presto 3.2 as an administrator.
2. Open a browser page with this URL, substituting the application server and port where Presto 3.2 is hosted:

   `http://`*app-server:port*`/presto/dev`

**Important:** This is an internal Presto tool that exposes internal APIs. You should not use these APIs unless specifically instructed by Presto Technical Support.

3. Click the **Presto APIs** tab and expand the **Repository Service** folder.
4. Select the `deactivateService` method. The content pane exposes a sample request for this API in JUMP format:




5. Change the string in the array value for the `params` property to `"MacroMetadataService"` and click **Execute**.
6. Select the `unregisterService` method.
7. Change the string in the array value for the `params` property to `"MacroMetadataService"` and click **Execute**.
8. Restart the Presto Server to register the MacroMetadataService for version 3.2. See "Start and Stop the Presto Server" on page 20 for instructions.

## Migrate Mashups using RAQL for Presto 3.6.1

Changes in the semantics in how datasets are stored in and retrieved from Presto Analytics In-Memory Stores between version 3.6 and 3.6.1 require updates to mashups that use the `<storeto>` or `<loadfrom>` extension statements.

- To keep the existing semantics, add a `version="1.0"` attribute to each `<storeto>` or `<loadfrom>` statement in your existing mashups.

  For example:

  ```
  <storeto version="1.0" key="stocks2011" variable="stocksDS"/>
  ...
  <loadfrom version="1.0" key="stocks2011"
      variable="stocks"/>
  ```

- To migrate to the new 3.6.1 semantics for existing mashups, you must:
  - Use a unique In-Memory Store name in the `cache` attribute to identify where each dataset should be stored or retrieved from.
  - Specify the method to assign unique keys when a dataset is stored.
  - In the previous release overwriting existing data was the default requiring explicit markup to have datasets appended to existing data. In this release, appending data is now the default, requiring explicit markup to clear existing data instead. To ensure mashups keep their existing semantics:

    Add `clearcache = "true"` to any existing mashup that uses the `<storeto>` statement but does *not* use the `append` attribute.

    Remove the `append` attribute from any `<storeto>` statements.

  For example:

  ```
  <storeto cache="myStocks2011" key="#unique"
      variable="stocksDS" clearcache="true"/>
  ...
  <loadfrom cache="myStocks2011" variable="stocks"/>
  ```

## Upgrade from Presto 3.6.1 to 3.7

The 3.7 release includes changes to the schema for the Presto Repository to provide better support for international character sets and user locales. These schema changes are *not* backwards compatible with previous releases, but they are only required if you need the internationalization features add in this release.

Because of this, there are two options you may choose from in upgrading from 3.6.1 to 3.7:

- If you do not require the internationalization features, you can perform a minimal upgrade, using your existing Presto Repository.

- If internationalization features are required, you must create a new Presto Repository and migrate all existing artifacts and configuration to the new release.

The steps involved in each of these upgrade paths are shown in the following table:

| Minimal Upgrade Without Internationalization | Major Upgrade With Internationalization |
|---|---|
| Follow the instructions for "Upgrade to New Versions of Presto and Migrate Artifacts" on page 211 keeping these points in mind:<br>- Mashups that use RAQL may need manual updates because of syntax changes in this release. See "Migrate Mashups That Use RAQL for 3.7" on page 225 for instructions.<br>- The 3.7 release has added the Integrated MashZone Server and the Event Service that allow users to work with events from the Software AGEvent Bus. To use these features, you must create a repository for MashZone and add configuration for both of these servers.<br>For instructions, see:<br>  - "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24<br>  - "Integrated MashZone Server Configuration and Administration" on page 147<br>  - "Event Service Configuration and Administration" on page 159<br>- Change the Presto base help URL to `http://mdc.jackbe.com/prestodocs/v3.7/` | 1. Install Presto 3.7.<br>2. Complete the required post-installation steps for Presto including creating a new repository for Presto and the Integrated MashZone Server.<br>See "Getting Started with the Presto Server" on page 13 for instructions.<br>3. Complete any optional configuration needed. See "What's Next" on page 50 for links.<br>See "Support International Character Sets and Locales" on page 85 for information on new configuration you may need for internationalization.<br>4. Migrate all artifacts from your existing Presto Repository to the new Presto Repository including any related extensions.<br>Contact your Software AG representative for assistance with this step.<br>5. Mashups that use RAQL may need manual updates because of syntax changes in this release. See "Migrate Mashups That Use RAQL for 3.7" on page 225 for instructions.<br>6. Change the Presto base help URL to `http://mdc.jackbe.com/prestodocs/v3.7/` |

## Migrate Mashups That Use RAQL for 3.7

In release 3.7, several updates were made to the Real-Time Analytics Query Language to provide closer compatibility with the SQL 2003 standard. This includes both changes to syntax and changes to some built-in functions. These changes are not backwards compatible.

To migrate to the 3.7 release, you may need to manually update mashups with RAQL statements in the following cases:

| Mashups with | May Need Manual Updates |
|---|---|
| <storeto> or <loadfrom> | RAQL now stores each dataset in its own In-Memory Store rather than storing many datasets in one In-Memory Store. This has significantly changed the syntax and semantics of both the <storeto> and <loadfrom> statements.<br><br>See "Handle In-Memory Store and Load Changes" on page 227 for migration techniques. |
| Column or function names that match reserved keywords | Many keywords from SQL are now reserved keywords for RAQL queries. Column or function names that match reserved keywords must be delimited to prevent errors when queries are run.<br><br>See "Handle Column or Function Name Incompatibilities" on page 227 for details and migration techniques. |
| Aggregate functions | It was possible in the previous release to include columns in the Select clause that were not used for grouping or for aggregate calculations when the query had a Group By clause or performed aggregation over the entire dataset. This is no longer valid.<br><br>**Note:** This does not apply to the use of aggregate functions with partitions or windows.<br><br>See "Updates for Non-Grouping Columns in Queries with Aggregation Functions" on page 228 for migration techniques. |
| Aliases in the Select clause | It was possible in the previous release to refer to an alias defined earlier in the Select clause within |

| Mashups with | May Need Manual Updates |
|---|---|
| | the definition for a subsequent column. This is no longer valid.<br><br>See "Updates for Alias References in Select Clauses" on page 229 for migration techniques. |
| No schema information | In some cases, errors can occur when queries involve datasets that do not have schema information to define their structure and datatypes. See "Handle Errors from Derived Schemas" on page 230 for more information. |
| Use of quote marks (") versus apostrophes (') | Quote marks and apostrophes are no longer interchangeable:<br><br>■ Quote marks are used to delimit names in query content that conflict with reserved words or that contain invalid characters.<br><br>■ Apostrophes are used to delimit literal values. |
| Small changes in valid function names | Function names should *not* include either periods (.) or dashes (-).<br><br>A single period is supported as the delimiter between the name of the library that the function belongs to and the name of the function. For example:<br><br>`myUDFLibrary.my_first_function`<br><br>You can include periods or dashes in function names if you enclose their name in quote marks. For example:<br><br>`myUDFLibrary."my-first-function"` |
| Small changes in valid column names | Column names should *not* contain periods (.) or dashes (-). You can, however, enclose invalid names in quote marks to handle this limitation.<br><br>See the technique described previously for function names for details. |
| Case sensitivity for dataset variables, column names and function names. | In queries, the variable names that identify a dataset are case sensitive just as any variable name is in EMML. However, column and function names are *not* case sensitive. |

## Handle In-Memory Store and Load Changes

RAQL supports both the original and the new semantics using a new `version` attribute. Mashups from previous releases that use either of these statements *must* be updated to indicate which syntax and semantics should be used:

- To continue to use your existing mashups and In-Memory Stores from 3.6, add `version="1.0"` to your existing mashups that use <storeto> or <loadfrom>. For example:

  ```
  <storeto key='stocks2011' variable='stocksDS'
      version='1.0'/>
  ```

- To update your existing mashups to use individual stores for each dataset you do not need to add the `version` attribute as the new syntax is the default. Instead, you must provide the name of the In-Memory Store to use with the `cache` attribute. You must also update the `key` attribute for <storeto> and remove it from <loadfrom>. The `scope`, `append` and `partionsize` attributes are not available.

## Handle Column or Function Name Incompatibilities

Many keywords from the SQL 2003 standard are now considered reserved words for RAQL.

Column or function names that match any of these reserved words, regardless of case, will cause errors in query execution. To avoid errors, you must enclose these conflicting names in quote marks.

In the following example, one column name conflicts with the date reserved word and must be quoted:

```
select company, item, "date", quantity from dailySales
```

Or this example, where the built-in month function must be quoted:

```
select symbol, open, close, "month"(quote_date) as quoted from stocks
```

Names may also include invalid characters such as periods, dashes or other punctuation characters. Enclosing them in quotes prevents query execution errors, such as this example:

```
select "current.balance" as curbal, "prev.balance" as prevbal from
accounts
```

Several of the built-in RAQL functions from the previous release now have name conflicts with reserved words, particularly those dealing with components in dates and times. You can update queries to enclose these functions in quotes, as shown previously. Or you can use the newly recommended functions that have no conflicts such as:

```
select symbol, open, close, extract_month(quote_date) as quoted from
stocks
```

The recommended functions that are available in this release for date/time built-in functions includes:

| 3.6 Function | 3.7 Recommended Function |
|---|---|
| date | extract_date |
| day | extract_day |
| hour | extract_hour |
| minute | extract_minute |
| month | extract_month |
| second | extract_second |
| week | extract_week |
| year | extract_year |

## Updates for Non-Grouping Columns in Queries with Aggregation Functions

In SQL, when you use aggregation functions with the entire dataset or with groups defined in a Group By clause, the columns the query can use in the Select clause can only be columns that are:

- Used to define groups, if any
- Used in an aggregate calculation

**Note:** This restriction is required to ensure valid data in the query result. Queries in this case return either one row for the entire dataset or one row for each group. A single aggregate result is returned for the functions for each result row. Columns used to define the groups are also guaranteed to have the same value for all dataset rows in a group, and thus have a single known value for the result row for that group. With other columns, however, a single known value is not guaranteed.

Note that this restriction does not apply when aggregate functions are used with partitions or windows as each row of the dataset is included in the query result. Aggregate calculations are simply added as new columns to each row.

RAQL did not enforce this restriction in the previous release. Thus the following query was legal:

```
select symbol, company, year(date) as yr,
    month(datetime) as mnth, avg(high) as average
    from stocks
```

```
    group by symbol, year(datetime), month(datetime)
```

In this case, the `company` column in Select is neither used in a calculation nor used in the group definition.

If you have queries with aggregation functions and non-grouping columns selected, you must correct them. In most cases, the solution is simply to remove the offending column. For the previous example, the correct query would be:

```
select symbol, extract_year(datetime) as yr,
    extract_month(datetime) as mnth, avg(high) as average
    from stocks
    group by symbol, extract_year(datetime),
    extract_month(datetime)
```

In rare cases such as the previous example, the `company` column which violates this rule also has a guaranteed single value for the corresponding `symbol` column used for grouping. Because of this, the `company` column can simply be added to the Group By clause without causing errors:

```
select symbol, company, extract_year(datetime) as yr,
      extract_month(datetime) as mnth, avg(high) as average
    from stocks
    group by symbol, company, extract_year(datetime),
      extract_month(datetime)
```

## Updates for Alias References in Select Clauses

In the previous release, you could define column aliases in the Select clause and refer to that alias in subsequent column definitions within that same clause. For example:

```
select symbol, open, close, (open + close) as expr1,
  (expr1 / 2) as expr2 , (expr2 * 100) as expr3
  from stocks
```

The definitions of the `expr2` and `expr3` columns use references to the `expr1` and `expr2` columns that are aliases defined earlier in this same Select clause.

This is not legal for SQL and is no longer supported in RAQL. If you have alias references within the same Select clause where the alias is defined, you must correct the query.

There are two methods to correct this syntax:

- Replace the illegal reference to the alias with an exact duplicate of the expression for this alias. From the previous example, this would change the query to:

  ```
  select symbol, open, close, (open + close) as expr1,
    ((open + close)/2) as expr2,
    (((open + close)/2) * 100) as expr3
    from stocks
  ```

- Use subqueries to define the alias in an earlier step.

  So another option for the previous example would be:

  ```
  select symbol, open, close, expr1, expr2, (expr2 * 100) as expr3
    from (
           select symbol, open, close, expr1, (expr1 / 2) as expr2
           from (
                   select symbol, open, close, (open + close) as expr1
  ```

```
                from stocks
            )
        )
```

### Handle Errors from Derived Schemas

In the previous release, RAQL treated data from datasets that did not include a schema as untyped. In 3.7, the RAQL Engine now derives a schema based on the first data row in the dataset.

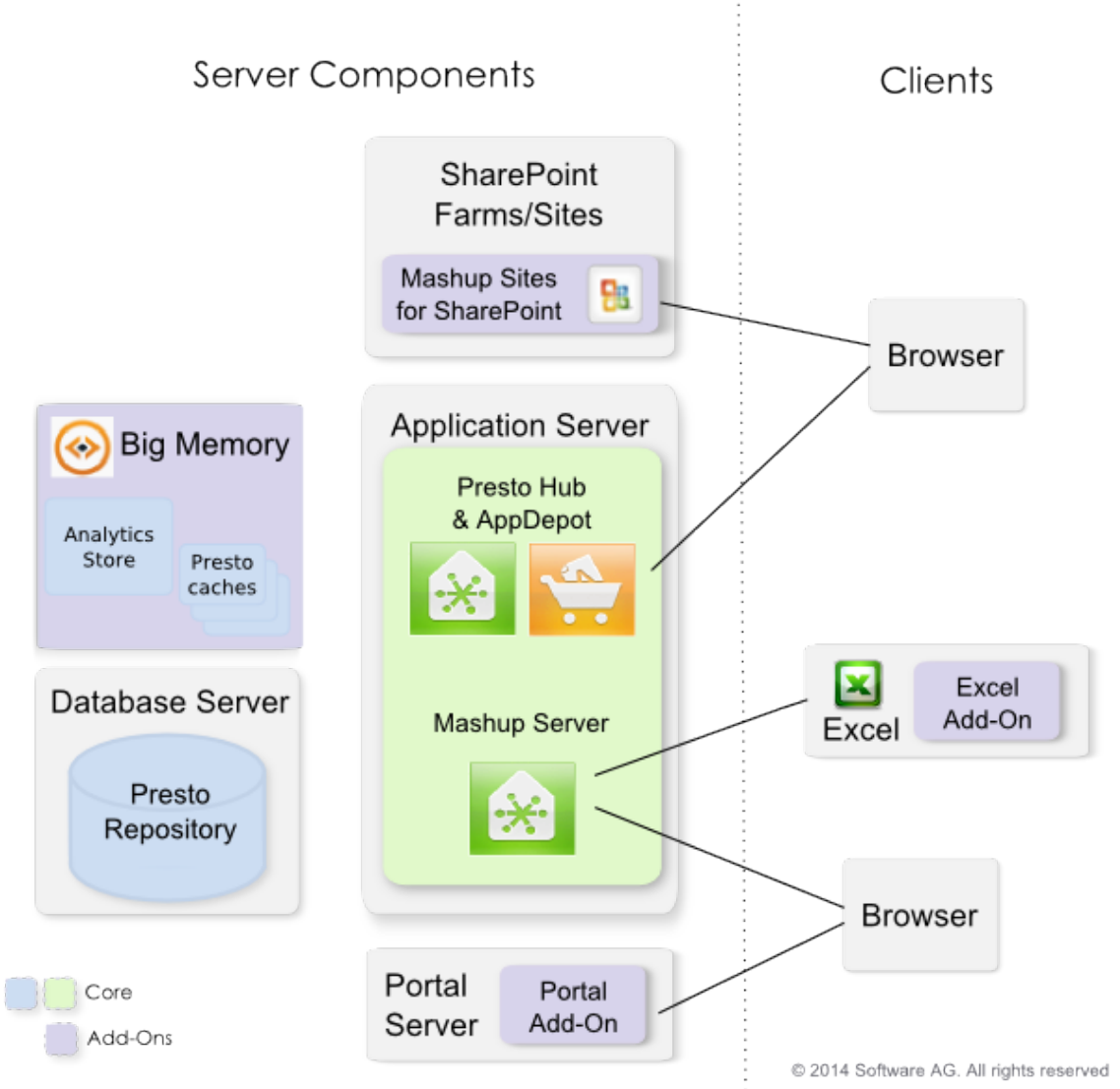
In most cases where a schema is not available, the datatype for columns is considered to be string. The data in the first row also defines the entire set of expected columns and the structure for all subsequent rows.

This derived schema can cause errors when the dataset is queried if the first row of the dataset is not a good representation of the dataset.

To prevent errors, the best practice is to provide schemas when datasets are loaded whenever possible.

## Deploying Presto Instances, Clusters or Artifacts

Deploying Presto to new hosts or new environments typically involves Deploying the Core Components, shown below, and optionally Deploying Server and Client Add-Ons or "Deploying Presto Artifacts and Other Metadata" on page 232.




## Deploying the Core Components

The core components include the Presto Server, Presto Hub and AppDepot (*web-apps-home* /presto), the Presto Repository, which is typically installed in a database other than the default Derby database, and the Presto Analytics In-Memory Stores and Presto caches.

**Note:** In earlier releases, the Presto Hub and AppDepot were deployed in a separate web application from the Presto web application. Effective in 3.2, all the core components are deployed in the single *web-apps-home* /presto web application.

For individual Presto servers, you typically do a default installation (see *Installing Software AG Products*). You may also move the Presto Repository to a database of your

choice. See "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 for instructions.

You can leave the the Presto Analytics In-Memory Stores and Presto caches in local memory for a single Presto 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 Presto Servers in One Host" on page 252. To deploy Presto servers in clusters, see "Clustering Presto Servers" on page 252 for requirements and links.

## Deploying Server and Client Add-Ons

You may also deploy two server-side add-ons: Presto Add-On for SharePoint and the Presto Add-On for Portals.

Users may also need to install the Presto Add-On for Excel, an add-on toolbar for Excel, to work with mashables and mashups in Microsoft Excel or to publish spreadsheets as mashables.

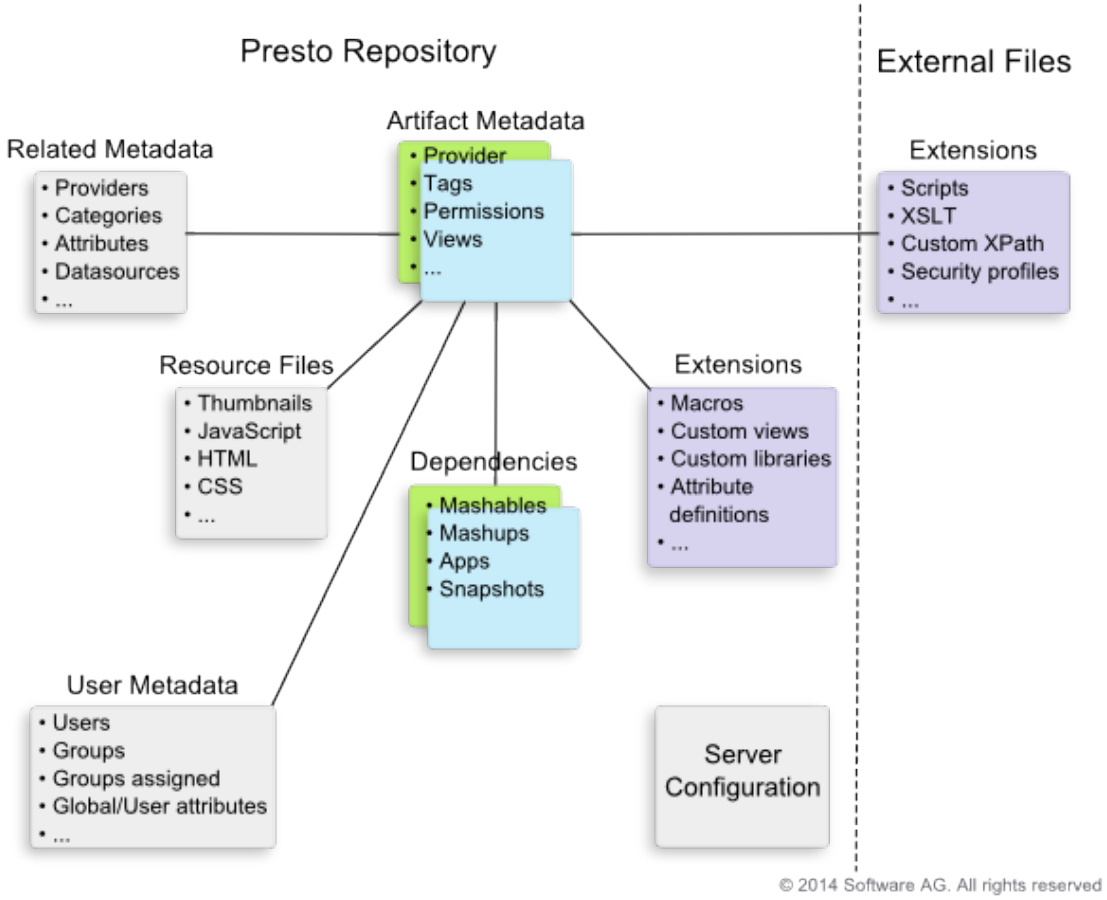
## Deploying Presto Artifacts and Other Metadata

You deploy specific artifacts (mashables, mashups or apps) and other metadata from a source Presto Server to a target Presto Server using the export and import commands.

**Important:** You *cannot* use export and import commands when the Presto version for the source and target Presto Servers are different:

- For major upgrades, use the migrate command instead. See "Upgrade to New Versions of Presto and Migrate Artifacts" on page 211 for details.
- 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 Presto Server and any macros, pluggable views or pluggable libraries that they may use.

   See the following topics for instructions using these Presto export commands:

   - "Exporting Mashable and Mashup MetaData" on page 236
   - "Exporting Macros" on page 238
   - "Exporting App MetaData" on page 239
   - "Exporting Pluggable Views or Libraries" on page 241
   - "Exporting Presto Global Attributes" on page 243
   - "Exporting Users, User Metadata and Groups" on page 244

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

**Table 5. Known Export/Import Limitations**

| | Exported | Not Exported |
|---|---|---|
| Artifact Metadata | ■ Basic metadata such as provider, category, tags and description.<br>■ Ownership (who created the artifact).<br>■ On/off status.<br>■ Run permissions.<br>■ Views.<br>■ Artifact attributes.<br>■ For apps, the AppDepot status. | ■ User rating and feedback.<br>■ Snapshots.<br>■ Snapshot schedules.<br>■ Caching configuration. |
| Related Metatdata/ User Metadata | ■ Providers.<br>■ Categories.<br>■ Global and user Presto attributes.<br>■ Users, groups and user group assignments if this data is tracked in the default Presto User Repository and not in your LDAP Directory.<br>■ Configuration for SharePoint connections used by SharePoint mashables or mashups. | ■ Datasources and their JDBC drivers that are used by database mashables or by mashups.<br>Datasources *must* be added to the target Presto Server before you import any artifacts that use them or the import will fail.<br>■ For apps that are published to the AppDepot, any user preferences for Favorite Apps.<br>■ User preferences for apps in Mashboard or the Mashboard state for workspace apps. |
| Resource Files | ■ Thumbnails for apps or pluggable views.<br>■ Screenshots for apps.<br>■ 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:<br>■ For workspace apps, this always exports all the apps used in the workspace.<br>■ For individual apps, you can choose to also export any mashables or mashups *explicitly declared and used* by those apps.<br>If you choose to include dependencies, all dependencies for basic apps are handled because Presto automatically declares dependencies for basic apps.<br>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.<br>■ For mashups, this always exports any other mashups or mashables that are used by the mashup.<br>■ 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.<br>■ 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.<br>■ Any of the file-based extensions such as custom XPath functions. See for a complete list. |

| Exported | Not Exported |
|---|---|
| Presto Server Configuration | Configuration for the Presto Server. |

2. Copy the files for any extensions used by the exported artifacts from the *presto-config* folder for the source Presto Server to the *presto-config* folder for the target Presto Server. See for a list of potential file-based extensions.

   **Note:** The *presto-config* folder may be an external configuration folder outside of the source and target Presto Servers or it may be in the default locations. See "Setting Up an External Presto Configuration Folder" on page 258 for more information on *presto-config* locations.

3. Define *datasources* in the Admin Console for the target Presto Server with *matching* names and JDBC drivers to the datasources in the source Presto Server. If these are not present, import of database mashables or mashups that use datasources fails.

   See "Manage Data Sources and Drivers" on page 131 for instructions.

4. If the exported artifacts include SharePoint mashables or mashups that use SharePoint as an information source, make sure the Presto license for the target Presto Server includes the Presto Add-On for SharePoint Add-On.

5. Use the export files created earlier to import mashables, mashups, apps, macros, pluggable views, pluggable libraries, Presto global and user attributes, users, groups and user group assignments from the source Presto Server.

   See the following topics for information on using these commands:

   - "Importing Macros" on page 246
   - "Importing Mashable or Mashup MetaData" on page 245
   - "Importing Pluggable Views or Libraries" on page 248
   - "Importing App Metadata" on page 247
   - "Importing Presto Global Attributes" on page 250
   - "Importing Users, User Metadata and Groups" on page 251

## Exporting Mashable and Mashup MetaData

Exporting mashables or mashups exports their metadata from one Presto Repository to a file. You can then import this file to another Presto 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 Presto Server.

**Important:** This command has specific limitations for what it exports. See "Table 5" on page 233 for details.

1. If it is not running, start the Presto Server for the Presto Repository that is the source for the mashables or mashups that you wish to export. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto 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 Presto Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
   - *-w password* : is the Presto 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 Presto 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:\Prestoversion\prestocli> padmin exportServices
  -q "type=REST" -f localRESTSvcs.xml -u Administrator -w manage
  >> localRestExport.log
```

## Exporting Macros

Exporting macros exports their metadata from one Presto Repository to a file. You can then import this file to another Presto 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 Presto Server. You can also use export and import for macros to make new custom blocks available in Wires for other instances of the Presto Server.

1. If it is not running, start the Presto Server for the Presto Repository that is the source for the macros that you wish to export. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto 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 Presto Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
- *-w password* : is the Presto 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 Presto 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 Presto 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 Presto Server at `presto12.myorg.com:8080` in the domain named `Finance`:

```
padmin exportEmmlMacro -f financeMacros.xml -d Finance -l presto12.myorg.com:8080 -u Administratc
```

This example exports all macros in custom domains from the Presto 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 Presto 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 Presto Repository. Typically, you export and import apps to move new apps to production or replicate data to a newly deployed Presto Server.

**Important:** This command has specific limitations for what it exports. See "Deploying Presto Artifacts and Other Metadata" on page 232 for details.

1. If it is not running, start the Presto Server for the Presto Repository that is the source for the apps you wish to export. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto 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 Presto Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
- *-w password* : is the Presto 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 Presto 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 Presto 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 Presto Repository. See the "Example 4. Examples" on page 243 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 Presto 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 Presto 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 Presto Server for the Presto Repository that is the source for the pluggable views or shared libraries that you wish to export. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto 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 Presto 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 Presto.

**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 Presto 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 Presto Server to export views and libraries from. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.

  Use this option if the Presto Server is remote, if it is not running in Tomcat or if it not using the default Tomcat port.
- *-u username* : the Presto username to use as credentials to execute this command. This account *must* have Presto administrator permissions.
- *-w password* : the Presto 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 Presto Server to a file named `localCustomViews.zip` that can then be used to deploy these libraries to another Presto Server using the importLib command.

```
c:\Prestoversion\prestocli> padmin exportLib
  -q "all" -f localCustomLibs -u Administrator -w manage
```

This next example, exports only pluggable views from a remote Presto Server to a file named `remoteCustomViews.zip` that can then be used to deploy these libraries to the local Presto Server using the importLib command.

```
c:\Prestoversion\prestocli> padmin exportLib
  -q "type=view" -f remoteCustomViews
  -l http://204.87.1.110:8080/presto/edge/api -u Administrator -w manage
```

## Exporting Presto Global Attributes

You can export metadata for all global Presto Attributes from the Presto Repository to an export file. You can then import this file to another Presto Repository.

**Note:** This command is available only in Presto 3.2 or later.

1. If it is not running, start the Presto Server for the Presto Repository with the global Presto attributes that you wish to export. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
   - *-w password* : is the Presto 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 Presto Repository.

## Exporting Users, User Metadata and Groups

Yyou can export all users, user groups, user group assignments and Presto User Attributes from the Presto Repository to an export file. You can then import this file to another Presto Repository.

**Important:** If you have configured Presto to work with your LDAP Directory, this command *only* exports Presto User Attributes. Data for users, user groups and user group assignments resides in LDAP.

This command is available only in Presto 3.2 or later.

1. If it is not running, start the Presto Server for the Presto Repository with the user groups that you wish to export. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or if it is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
- *-w password* : is the Presto 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 Presto Repository.

## Importing Mashable or Mashup MetaData

you must have a mashable or mashup export file to import. See "Exporting Mashable and Mashup MetaData" on page 236 for instructions.

If the export file contains database mashables or mashups that use named datasources, you or a Presto 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 Presto Server that was the source of these mashables or mashups.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
- *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
- *-w password* : is the Presto 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 Presto 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:\Prestoversion\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 238 for instructions.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
   - *-p password* : is the Presto 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 Presto Hub.

## Importing App Metadata

you must have an app export file to import. See "Exporting App MetaData" on page 239 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 245 for more information.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.

- *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
- *-w password* : is the Presto 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 Presto 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 Presto Server. See "Exporting Pluggable Views or Libraries" on page 241 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 Presto Server to another Presto Server. See "Example 7. Example" on page 250 for an example of this usage.

You may also use this command to add to or update pluggable views or pluggable libraries Presto based on source files on your computer.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.

2. Open a command or terminal window and move to the *presto-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 Presto.

  **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 Presto 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 Presto Server where these views and libraries should be imported to. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.

Use this option if the Presto Server is remote, if it is not running in Tomcat or if it is not using the default Tomcat port.

- *-u username* : the Presto 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 Presto 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 Presto Server.

```
c:\Prestoversion\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 Presto Server.

```
c:\Prestoversion\prestocli> padmin importLib
  -q "subtype=network" -f remoteCustomLibs.zip -u Administrator -w manage
```

## Importing Presto Global Attributes

you must have a Presto Global Attribute export file to import. See "Exporting Presto Global Attributes" on page 243 for instructions.

**Note:** This command is available only in Presto 3.2 or later.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.

- *-p password* : is the Presto 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 Presto Global Attribute to overwrite an existing Presto 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 Presto Global Attributes have been imported in Presto 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 244 for instructions.

**Note:** This command is available only in Presto 3.2 or later.

A user export file contains Presto User Attributes. It may also contain users, user groups and user group assignments if you are using the default Presto 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 Presto Server has been changed to `welcome`.

1. If it is not started, start the Presto Server for the Presto Repository where you wish to import data. See "Start and Stop the Presto Server" on page 20 for instructions.
2. Open a command window and move to the *presto-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 Presto Server is remote or is not running in Tomcat on the default Tomcat port. If you omit this option, this defaults to `http://localhost:8080/presto/edge/api`.
   - *-u username* : is the Presto username to log in with. This account *must* have Presto administrator permissions.
   - *-p password* : is the Presto 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 Presto global attribute to overwrite an existing user, group, user group assignments or Presto 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 Presto Hub.

## Deploying Multiple Presto Servers in One Host

You can deploy several different, independent Presto Servers on a single host. Each Presto Server must be hosted in its own application server and have its own Presto Repository.

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

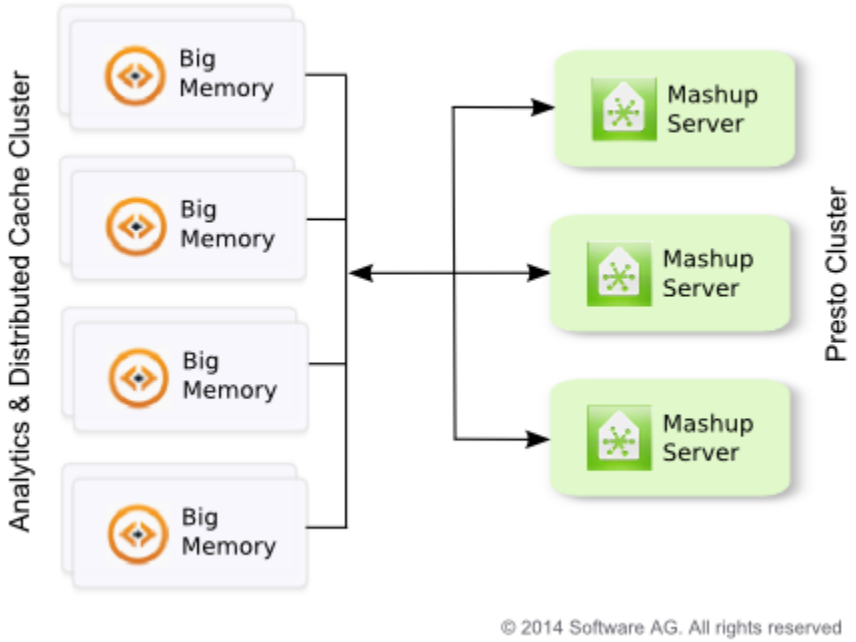
> **Note:** You can also create clusters of Presto Servers to provide load balancing. See "Clustering Presto Servers" on page 252 for information.

## Clustering Presto 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 Presto is a web application, using an HTTP session based on J2EE standards, you can apply the same cluster architectures and solutions to Presto that you use with other web applications.

> **Note:** Clustering is *not* supported for Presto if you also deploy Software AG MashZone or you use the Integrated MashZone Server with Presto.

A common architecture for Presto clusters looks something like this:




See "Setting Up a New Cluster" on page 253 or "Adding New Members to an Existing Cluster" on page 255 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 Presto Configuration Folder" on page 258: this allows you to keep most of the configuration and extensions for Presto 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 Presto Server in the cluster and complete most of the basic configuration that will be shared across the cluster.

- "Sharing the Presto Repository in Clustered Environments" on page 256: all nodes in the cluster work with a shared Presto Repository which you must create and configure.

Sharing the Presto 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 Presto Server has a local cache for mashable and mashup responses as well as local caches for updates to artifacts. If Presto Analytics is enabled in your Presto license, the Presto 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 Presto Servers in the cluster share.
  - *Must* configure a distributed cache for artifact updates that all Presto Servers in the cluster share.
  - *Must* configure a distributed cache for the Presto Analytics In-Memory Stores that all Presto Servers in the cluster share.

  See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for instructions on how to configure BigMemory, or other caching solutions, as a distributed cache for .Presto

- *Defining the Application Server Cluster*: the application servers that host each Presto 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, Presto 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 Presto 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 255 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 Presto Artifacts and Other Metadata" on page 232 for instructions.

## Adding New Members to an Existing Cluster

**To add additional Presto Servers to an existing cluster**

1. Install the Presto Server. See *Installing Software AG Products* for instructions.
2. Configure the Presto Server to use the shared Presto Repository for the cluster. See "Share an Existing Presto Repository" on page 257 for instructions.
3. If the cluster has a shared external configuration folder, add this folder and any subfolders to the classpath for the Presto 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 Presto Server in the cluster to the new Presto Server.

   See "Presto File-Based Configuration and Extensions" on page 260 for a list of files and folders to copy. Be sure to also copy the JDBC drivers used for database mashables and mashups. See "Sharing JDBC Drivers in Clustered Environments" on page 264 for more information.
5. Copy the server configuration that cannot be shared from an existing Presto Server in the cluster to the new Presto Server. See "Presto File-Based Configuration and Extensions" on page 260 for details on the files and locations for this step.
6. Update the application server that hosts the new Presto 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, Presto 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 Presto Server.

## Sharing the Presto Repository in Clustered Environments

In clustered environments, all Presto Servers in the cluster must work with a single, shared Presto Repository. You can "Create and Share a New Presto Repository" on page 256 with cluster members, typically when you are creating new environments. Or you can "Share an Existing Presto Repository" on page 257 within a cluster.

### Create and Share a New Presto Repository

**To create a new shared repository**

1. Create a new Presto Repository in the appropriate database for your environment.

   Using the SQL tool for the database that will be host, add Presto Repository tables with the scripts shown below from the corresponding folder in *presto-install* /prestorepository:

| Database | Folder | SQL Scripts |
|---|---|---|
| Microsoft SQL Server | mssqldb | ■ createDBTables.txt for MetaData and the default User Repository<br>■ createSnapsTables.txt for Snapshots<br>■ createSchedulerTables.txt for Scheduler |
| MySQL | mysqldb | |
| Oracle | oracledb | |
| PostgreSQL | postgresdb | |

   There are also scripts to drop the corresponding Presto Repository tables in these folders, if needed.

2. Copy the JAR file for the JDBC driver for your database to the *presto-config* folder for JAR files.

   If you are using a shared external configuration folder, you can copy the JAR to that single folder. For example, you might copy this to G:\PrestoConfig\lib on a Windows system or /mountA/home/PrestoConfig/lib on Linux, Mac OS X or UNIX systems.

   If you are not sharing a configuration folder across the cluster, you must copy the JAR to *web-apps-home* /presto/WEB-INF/lib for each Presto Server that uses this Presto Repository. See "Setting Up an External Presto Configuration Folder" on page 258 for more information.

3. If this is a new cluster, update configuration information for the MetaData, User and Snapshot repositories for one Presto Server in the cluster. See steps 3 onward

in "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 for instructions.

4. Copy the rdsJdbc.properties file to the *presto-config* folder for the entire cluster or to each Presto Server in the cluster.

   If you are using a shared external Presto 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 Presto Configuration Folder" on page 258 for more information.

5. Enable distributed caching for artifacts (required) and optionally distributed caching for mashable/mashup responses. See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for more information and instructions.

6. If the Presto Repository is hosted in Microsoft SQL Server, MySQL or Oracle, change the repository JAR in the Presto Server. See "Upgrade the Presto 3.5 Repository JAR for MSSQL, MySQL or Oracle" on page 216 for instructions.

7. Restart each Presto Server in the cluster.

## Share an Existing Presto Repository

If you are creating a cluster using an existing Presto Repository or simply adding members to an existing cluster, you simply update each new Presto Server in the cluster to use the existing repository.

1. If the cluster does *not* have a shared JDBC driver folder and a shared external configuration folder:

   a. Copy the JAR file for the JDBC driver for your database to the *web-apps-home* /presto/WEB-INF/lib folder for the new Presto Server cluster member.

   b. Copy the rdsJdbc.properties file from the *web-apps* /presto/WEB-INF/classes folder of an existing Presto Server in the cluster, to the same folder for the new cluster member.

   See "Setting Up an External Presto Configuration Folder" on page 258 for more information on shared configuration for clusters.

2. If the cluster does have a shared JDBC driver folder and a shared external configuration folder, update the application server's classpath for the new cluster member to include all three folders.

   Depending on your application server, you may update the classpath in the administration console, in configuration files or in the startup script for the application server. See documentation for your application server for more information.

   For Tomcat on Windows, for example, you can edit the *tomcat-install* /bin/catalina.bat file and add the path to this folder to the classpath environmental variable to be something like this:

   ```
   set "CLASSPATH=%CLASSPATH%;C:\PrestoConfig;C:\PrestoConfig\classes;C:
   \PrestoConfig\lib;C:\PrestoConfig\db\jdbc"
   ```

On Linux, Mac OS X or UNIX systems, you would update *tomcat-install* /bin/catalina.sh to something like this:

```
CLASSPATH="$CLASSPATH":/users/PrestoConfig:/users/PrestoConfig/
classes:/users/PrestoConfig/lib:users/PrestoConfig/db/jdbc
```

3. Enable distributed caching for artifacts (required) and optionally distributed caching for mashable/mashup responses. See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for more information and instructions.
4. If the Presto Repository is hosted in Microsoft SQL Server, MySQL or Oracle, change the repository JAR in the Presto Server. See "Upgrade the Presto 3.5 Repository JAR for MSSQL, MySQL or Oracle" on page 216 for instructions.
5. Restart the new Presto Server for this cluster.

## Setting Up an External Presto Configuration Folder

Most configuration for Presto and most of the extensions that you add for your organization's use are stored in the Presto Repository. However, some Presto configuration and extensions are file based.

By default, Presto keeps configuration and extensions in the Presto Server web application in these folders:

- *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/classes for class, configuration and extension files
- *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/lib and *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/config for JAR files.

You can move most of these configuration and extension files to folders that are external to the Presto Server.

**Important:** Presto documentation refers to all of these folders as *presto-config* .

Using external configuration folders for Presto is a best practice as they simplify deployment and upgrades of the Presto Server. They also simplify configuration management for clustered environments. External configuration folders are not required, however.

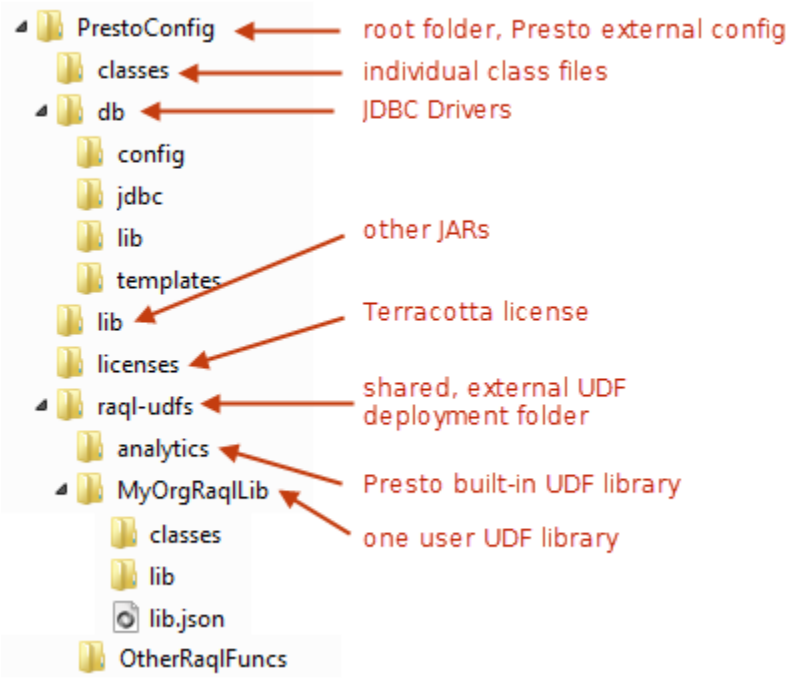
**To create and use an external configuration folder for Presto**

1. Create the top-level external folder to use for Presto configuration, such as PrestoConfig. In clustered environments, share or mount this folder across the entire cluster.

   You can create subfolders under this external folder to organize configuration and extensions.
2. For clustered environments, create subfolders under the top-level external configuration folder for:

- The standard classes and lib folders.
- JDBC drivers for database mashables and mashups for the cluster. If desired, you can create an external folder for JDBC drivers in non-clustered environments. See "Sharing JDBC Drivers in Clustered Environments" on page 264 for instructions.
- Built-in and user-defined functions for use in RAQL queries for Presto Analytics.

The external configuration folder tree should now look something like this:




3. If not complete, finish configuration for the Presto Server and move the configuration and extension files to the external configuration folder or an appropriate subfolder. See the "Presto File-Based Configuration and Extensions" on page 260 section for the specific configuration steps, files and locations.
4. Add the external Presto configuration folder, *and any subfolder* that contains extensions or JAR files, to the classpath for the application server(s) hosting the Presto Server.

   You may update the classpath in configuration files or in the startup script for the application server.

   For Windows environments, for example, you can edit the *tomee-install* /bin/setenv.bat file and update the classpath environmental variable to be something like this:

   ```
   set "CLASSPATH=%CLASSPATH%;C:\PrestoConfig;C:\PrestoConfig\classes;C:\PrestoConfig\lib;C:\PrestoConfig\db\jdbc"
   ```

On Linux, Mac OS X or UNIX systems, you would update *tomee-install* /bin/setenv.sh to something like this:

```
CLASSPATH="$CLASSPATH":/users/PrestoConfig:/users/PrestoConfig/
classes:/users/PrestoConfig/lib:users/PrestoConfig/db/jdbc
```

## Presto File-Based Configuration and Extensions

Most file-based configuration or extensions involve information that Presto needs to connect to the Presto 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 "Presto Configuration Files That Can Be External" on page 260 and "Presto Extensions" on page 263 for details on resources that can be shared across a cluster.

Some file-based configuration, however, *must* reside in the web application for each Presto Server. In clusters, this configuration must be replicated in each cluster member. See "Presto Configuration Files That Must Be Internal " on page 261 for details.

### Presto Configuration Files That Can Be External

| File | Description and Configuration | Default Location |
|---|---|---|
| dynamiccache.xml | Default configuration information for dynamic In-Memory Stores created by Presto Analytics. | *presto-install* / apache-tomee-jaxrs/presto/WEB-INF/classes |
| ehcache.xml | Configuration information for Presto caches. This also contains configuration for Presto Analytics In-Memory Stores from version 3.6. | |
| presto.config | Miscellaneous Presto properties, including the path to the deployed web app home folder and the path, if used, for shared JDBC drivers. See "Sharing JDBC Drivers in Clustered Environments" on page 264 for more information. | |
| rdsJdbc.properties | Connection information for the MetaData and User section of the Presto Repository. | |

| File | Description and Configuration | Default Location |
|---|---|---|
| | **Note:** Although Presto uses JNDI to connect to the Presto Repository, JDBC connection properties are also used in some specific cases.<br><br>See "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 for details. | |
| The Terracotta BigMemory license file | The license file for BigMemory, used for Presto caches and Presto AnalyticsIn-Memory Stores, is a separate license file from the Presto license. You can keep the BigMemory license in an external folder shared across the cluster.<br><br>See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for required configuration steps to enable a shared license. | |
| userRespositoryLdap.properties | Connection information for your LDAP Directory. See "Integrate Your LDAP Directory with Presto" on page 40 for details. | |

#### Presto Configuration Files That Must Be Internal

The file-based configuration that must remain in each Presto Server web application resides in the *web-apps-home* /presto/WEB-INF/classes folder.

For upgrades to new Presto versions, you can generally copy these configuration files from your existing Presto version to the new version. Review the Presto 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 Presto 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 Presto. See "Configure BigMemory Servers for Presto Caching and In-Memory Stores" on page 117 for more information.<br><br>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 Presto. See "Authentication with Single Sign-On Solutions" on page 59 or "Authentication with Digital Certificates/SSL" on page 66 for more information. |
| applicationContext-security-x509.xml | You edit this file initially to enable X509 certificate authentication for Presto. See "Authentication with Digital Certificates/SSL" on page 66 for more information. |
| applicationContext-scheduler.xml | You edit this file when you move the Presto Repository from the default Derby database to a robust solution. See "Move the Presto and MashZone Repositories to a Robust Database Solution" on page 24 for more information. |
| log4j.properties | This file is updated automatically when you change logging configuration in the Admin Console. See "Configure Logging for the Presto Server" on page 104 for details.<br><br>When you change logging for Presto Servers in a cluster, only the specific Presto Server that the Admin Console |

| File | Description and Configuration |
|---|---|
| | 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 | You edit these files when you configure Presto to use your LDAP Directory as the user repository. See "Integrate Your LDAP Directory with Presto" on page 40 for details. |
| userRepositoryApplicationContext-ldap.xml | |

### Presto Extensions

Some extensions, such as macros, are registered and reside in the Presto Repository. Any of the following file-based extensions can reside in an external folder:

| File | Default Location |
|---|---|
| JAR files for JDBC drivers for datasources used with database mashables or with mashups.<br><br>To share JDBC drivers across a cluster of Presto Servers, you must set up a shared JDBC folder. See "Sharing JDBC Drivers in Clustered Environments" on page 264 for the specific details.<br><br>If you do not create a shared folder, then you *must* copy the JARs for all JDBC drivers to the default location for each Presto Server in the cluster. | *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/config/db/jdbc |
| Scripts or classes called in mashups in EMML using the <script> statement. This includes:<br><br>■ JavaScript files<br>■ Any Java class that is not in `java.lang`<br>■ JRuby scripts<br>■ Groovy scripts | *presto-install* /apache-tomee-jaxrs/presto/WEB-INF/classes<br><br>or<br><br>*presto-install* /apache-tomee-jaxrs/presto/WEB-INF/lib (for JARs) |
| XSLT stylesheets called in mashups in EMML using the <xslt> statement. | |

| File | Default Location |
|---|---|
| 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 70 for details. | |
| Custom filter classes for single sign-on authentication. See "Implementing a Custom SSO Filter" on page 65 for details. | |
| Classes and third-party libraries for a user-defined function library to use with RAQL. | |

## Sharing JDBC Drivers in Clustered Environments

In clustered environments, each cluster member must have access to the JDBC drivers that are used by database mashables or mashups. You can create a folder for JDBC driver JARs that is external to all Presto Servers in the cluster. This external folder can be shared with all cluster members to ensure access. Once you have set up a shared driver folder, any JDBC drivers that you add in the Admin Console are immediately available to all cluster members.

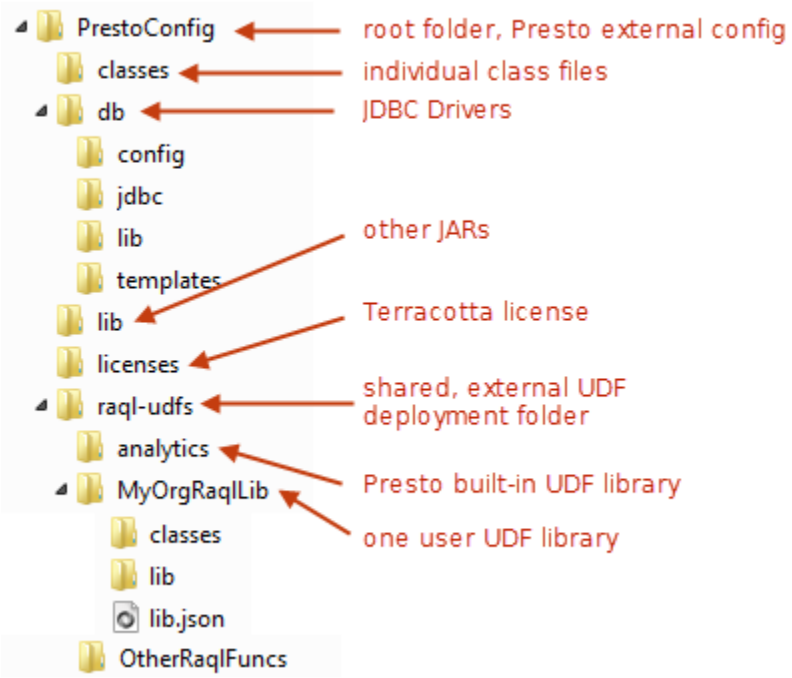
**Note:** If you do not set up a shared JDBC driver folder, you must copy the JARs for each driver, as they are added, to the default driver location for each Presto Server in the cluster.

1. Create a folder for JDBC drivers that is external to the Presto Server web application. In clustered environments, this folder should reside in a file system that is shared or mounted across the entire cluster.

   If you are using an external configuration folder for Presto, it is a best practice to add the JDBC driver folder there. See "Setting Up an External Presto Configuration Folder" on page 258 for more information.

2. Copy the entire *web-apps-home* /presto/config/db folder from one Presto Server in the cluster to the external shared folder for drivers created in the previous step.

Your shared folder tree should look something like this:




3. In any text editor, edit the *presto-config* /presto.config file.

> **Note:** The *presto-config* folder may be in the default location or in an external location. See "Setting Up an External Presto Configuration Folder" on page 258 for more information.

   a. Set the `codegen.absolutepath.config` property to point to the external JDBC driver folder.

      For a Windows system with the external JDBC driver folder on a shared drive G:\PrestoConfig\db, for example, the property would look like this:

      codegen.absolutepath.config = g:\\PrestoConfig

      For Linux, Mac OS X or UNIX systems with an external folder of /mountA/home/PrestoConfig/db, it might look something like this:

      codegen.absolutepath.config = /mountA/home/PrestoConfig

   b. Save the changes to this file.

4. Restart the entire cluster to apply these changes.