Oracle® Endeca Information Discovery Studio

Studio Administration and Customization Guide

Version 3.0.0 • March 2013

Copyright and disclaimer

Table of Contents

Preface

Oracle® Endeca Information Discovery Studio is an enterprise data discovery platform for advanced, yet intuitive, exploration and analysis of complex and varied data.

Information is loaded from disparate source systems and stored in a faceted data model that dynamically supports changing data. This integrated and enriched data is made available for search, discovery, and analysis via interactive and configurable applications. Oracle Information Discovery Studio includes a Provisioning Service that allows you to upload data directly from spreadsheet files.

Oracle Endeca Information Discovery Studio enables an iterative "model-as-you-go" approach that simultaneously frees IT from the burdens of traditional data modeling and supports the broad exploration and analysis needs of business users.

About this guide

This guide provides information on configuring, administering, and customizing Oracle Endeca Information Discovery Studio.

Who should use this guide

This guide is intended for administrators who need to configure and monitor Studio, and developers who want to extend and customize Studio.

Conventions used in this document

The following conventions are used in this document.

Typographic conventions

The following table describes the typographic conventions used in this document.

Typeface	Meaning
User Interface Elements	This formatting is used for graphical user interface elements such as pages, dialog boxes, buttons, and fields.
`Code Sample`	This formatting is used for sample code phrases within a paragraph.
Variable	This formatting is used for variable values. For variables within a code sample, the formatting is *`Variable`*.
`File Path`	This formatting is used for file names and paths.

Symbol conventions

The following table describes symbol conventions used in this document.

Symbol	Description	Example	Meaning
>	The right angle bracket, or greater-than sign, indicates menu item selections in a graphic user interface.	File > New > Project	From the File menu, choose New, then from the New submenu, choose Project.

Contacting Oracle Customer Support

Oracle Customer Support provides registered users with important information regarding Oracle software, implementation questions, product and solution help, as well as overall news and updates from Oracle.

You can contact Oracle Customer Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Part I

Configuring and Monitoring Studio

Chapter 1

Configuring Framework Settings

Framework settings are used to configure state, security, and other settings for Studio.

About the Framework Settings page

The **Framework Settings** page on the **Control Panel** allows you to view and configure the framework settings.

Note: If you do not see the **Framework Settings** option in the **Control Panel** menu, it probably means you did not install the `endeca-framework-settings-portlet-<`*`version`*`>.war` file. Please review your installation settings.

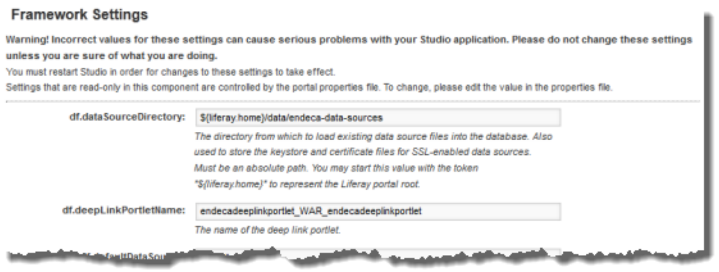
The default values of these settings are created automatically upon first use. You can only edit the settings. You cannot add or delete settings.

Settings only appear after the feature(s) that use them have been executed at least once.

Settings that have been configured in `portal-ext.properties` are displayed, but cannot be edited.

Configuring Framework Settings from the Control Panel

You use the fields on the **Framework Settings** page to modify the settings. You cannot modify settings that already have been configured in `portal-ext.properties`. If a setting has been configured directly in the `portal-ext.properties` file, then the field on the **Framework Settings** page is locked.



Framework Settings

Warning! Incorrect values for these settings can cause serious problems with your Studio application. Please do not change these settings unless you are sure of what you are doing.

You must restart Studio in order for changes to these settings to take effect.

Settings that are read-only in this component are controlled by the portal properties file. To change, please edit the value in the properties file.

df.dataSourceDirectory:	${liferay.home}/data/endeca-data-sources *The directory from which to load existing data source files into the database. Also used to store the keystore and certificate files for SSL-enabled data sources. Must be an absolute path. You may start this value with the token "${liferay.home}" to represent the Liferay portal root.*
df.deepLinkPortletName:	endecadeeplinkportlet_WAR_endecadeeplinkportlet *The name of the deep link portlet.*

The **Framework Settings** page contains the following settings:

Setting	Description
df.dataSourceDirectory	The directory from which existing data source files will be uploaded into the database. This directory also is used to store keystore and certificate files for secured data sources.
df.deepLinkPortletName	The name of the deep link component.

df.defaultChartColorPalette	The default set of colors to use to display charts in the **Chart** component.
	
	The value is a comma-separated list of 16-30 hex color values.
	
	For reference, the default value is:
	
	`#57BCC1,#F3A900,#A5C500,#9C2E5B,#C4B25D,#0072B1,#229903,#D55E00,`
	`#F2D900,#A279CD,#ABDEE0,#AA7600,#D2E280,#6D2040,#E2D9AE,#00507C,`
	`#91CC81,#954200,#F9EC80,#71548F,#3D8387,#F9D480,#738A00,#CE97AD,`
	`#897C41,#80B9D8,#186B02,#EAAF80,#A99700,#D1BCE6`

df.defaultDataSource	The name of the data source to use as the default.
df.defaultExporter	The default exporter class.
df.exportPortletName	The default name of the export portlet.
df.healthCheckTimeout	The time (in milliseconds) for query timeout when checking data source availability on initialization.
df.mapLocation	The URL for the Oracle MapViewer eLocation service. The eLocation service is used for the text location search on the **Map** component, to convert the location name entered by the user to latitude and longitude. By default, this is the URL of the global eLocation service. If you are using your own internal instance, and do not have Internet access, then set this setting to "None", to indicate that the eLocation service is not available. If the setting is "None", Studio disables the text location search. If this setting is not "None", and Studio is unable to connect to the specified URL, then Studio disables the text location search. Studio then continues to check the connection each time the page is refreshed. When the service becomes available, Studio enables the text location search.

Setting	Description
df.mapTileLayer	The name of the MapViewer Tile Layer. By default, this is the name of the public instance. If you are using your own internal instance, then you must update this setting to use the name you assigned to the Tile Layer.
df.mapViewer	The URL of the MapViewer instance. By default, this is the URL of the public instance of MapViewer. If you are using your own internal instance of MapViewer, then you must update this setting to connect to your MapViewer instance.
df.maxExportBaseRecords	The maximum allowable number of non-EQL records that can be exported.
df.maxExportEQLRecords	The maximum allowable number of EQL records that can be exported.
df.mdexCacheManager	The fully-qualified class name to use for the MDEX Cache Manager. Changing this setting is currently experimental and unsupported, and should be used only for research purposes. This interface will change in upcoming releases.
df.mdexSecurityManager	The fully-qualified class name to use for the MDEX Security Manager.
df.mdexStateManager	The fully-qualified class name to use for the MDEX State Manager.
df.metadataCacheEnabled	Whether to use the metadata cache. The default value is `true`, indicating that whenever Studio needs attribute metadata, it retrieves it from the metadata cache. If this setting to `false`, then whenever Studio needs attribute metadata, it retrieves it directly from the data source, which is usually slower. Any change to this setting takes effect for new sessions. Note that the cache only becomes out of synch with the data source if you use the Endeca Server web services to update the metadata manually, for example using Integrator or a custom application. If you update the metadata from within Studio, the cache is updated automatically. So unless you know you will be making manual updates, you should leave the setting set to `true`. If the cache does get out of synch, then to update the cache: 1. Go to the **Attribute Settings** page. 2. From the drop-down list, select the data source. 3. Click **Reload Attribute Cache**.

Setting	Description
df.provisioningServiceLimit	The maximum number of data domains created by the Provisioning Service that can be present on the Endeca Server. Once this limit is reached, the file upload option becomes unavailable until one or more of those data domains is removed. To indicate that there is no limit, set this value to 0.
df.wsConnectionTimeout	The time in milliseconds before a connection to the Conversation, Configuration, Entity Configuration, or EQL Parser web service times out. The default value is 300000. If these connections are timing out frequently, then some possible causes are: • The Endeca Server is overloaded, and might benefit from upgraded hardware. • A problem in the networking hardware is causing bottlenecks.

On the **Framework Settings** page, to change a setting:

1. Provide a new value in the setting configuration field.

   Note: Take care when modifying these settings, as incorrect values can cause problems with your Studio application.

   If the setting is configured in `portal-ext.properties`, then you cannot change the setting from this page. You must set it in the file.
2. Click **Update Settings**.
3. To apply the changes, restart Studio.

Configuring Framework Settings in portal-ext.properties

By default, you configure settings from the **Framework Settings** page. You also can add one or more of the settings to the `endeca-portal\portal-ext.properties` file.

Configuring settings in `portal-ext.properties` makes it easier to migrate settings across different environments. For example, after testing the settings in a development system, you can simply copy the properties file to the production system, instead of having to reset the production settings manually from the **Control Panel**.

In the file, the format for adding a setting is:

```
<settingname>=<value>
```

Where:

- <*settingname*> is the name of the setting from the **Framework Settings** page.
- <*value*> is the value of the setting.

For example, to set the default data source in the file, the entry would be:

```
df.defaultDataSource=mydefault
```

If a property is configured in `portal-ext.properties`, you cannot edit it from the **Control Panel**. The field on the **Framework Settings** page is read only.

To move the configuration for a setting to the properties file after Studio has been started:

1. Stop the server.
2. Add the setting to `portal-ext.properties`.
3. Restart Studio.

   On the **Framework Settings** page of the **Control Panel**, the setting is now read only. You can no longer edit the value from the **Control Panel**.

Chapter 2

Configuring Logging for Studio

Studio logging helps you to monitor and troubleshoot your Studio application.

About logging in Studio

Studio uses the Apache log4j logging utility.

The Studio log files include:

- A main log file with most of the logging messages
- A second log file for performance metrics logging

You can also use the **Performance Metrics** page of the **Control Panel** to view performance metrics information.

For more information about log4j, see the *Apache log4j site*, which provides general information about and documentation for log4j.

About the log4j configuration XML files

The primary log configuration is managed in `portal-log4j.xml`, which is packed inside the portal application file `WEB-INF/lib/portal-impl.jar`.

To override settings in `portal-log4j.xml`, you use the file `portal-log4j-ext.xml`, which is located in the portal application's `/WEB-INF/classes/META-INF/` directory.

Both files are in the standard log4j XML configuration format, and allow you to:

- Create and modify appenders
- Bind appenders to loggers
- Adjust the log verbosity of different classes/packages

By default, `portal-log4j-ext.xml` specifies a log verbosity of INFO for the following packages:

- `com.endeca`

- `com.endeca.portal.metadata`
- `com.endeca.portal.instrumentation`

It does not override any of the default log verbosity settings for non-Information Discovery components.

Note: If you adjust the logging verbosity, it is updated for both log4j and the Java Utility Logging Implementation (JULI). Code using either of these loggers should respect this configuration.

About the Studio log files

For Studio, one log file contains all of the log messages, and a second file is used only for metrics logging.

About the main Studio log file

In the Studio log file configuration, the main root logger prints all messages to:

- The console, which typically is redirected to the application server's output log (For Tomcat, `catalina.out` and for WAS, `SystemOut.log`)
- A file called `eid-studio.log`

The main logger does not print messages from the `com.endeca.portal.instrumentation` classes. Those messages are printed to the metrics log file.

Location of eid-studio.log

By default, the logger tries to create `eid-studio.log` in the following directory:

App Server	Default Log File Location

Tomcat:	The default location of `eid-studio.log` is:
	`<`*`value of catalina.home`*`>/logs`
	For example, if `catalina.home` is set to `C:\endeca-portal\tomcat-6.0.29`, then `eid-studio.log` is located in:
	`C:\endeca-portal\tomcat-6.0.29\logs`

WebLogic 11g:	The default location of `eid-studio.log` is the root directory of the WebLogic domain.

If the logger can't place the file in the default directory, then you typically can find `eid-studio.log` in one of the following locations:

App Server Configuration	Alternate Log File Location
Tomcat - startup script:	If you started Tomcat by running a startup script, the log file is located where the script was run. For example, if you ran the startup script from `tomcat-<version>/bin`, the log file also is in `tomcat-<version>/bin`.

App Server Configuration	Alternate Log File Location
Tomcat - Windows service:	If you registered and started Tomcat as a Windows service, the log file may be in `C:\Windows\System32` or `C:\Windows\SysWOW64`.
Tomcat - Eclipse server:	If Tomcat is a server inside of Eclipse, the log files may be located in the root of the Eclipse directory.
WebLogic 11g:	Does not apply. The log file is always in the domain directory.

Specifying an absolute path to the log file

The custom appender used for Studio logging contains the logic for determining the log file directory.

```
<appender name="FILE" class="com.endeca.portal.util.ContainerAwareRollingFileAppender">
    <param name="File" value="eid-studio.log"/>
    <param name="MaxFileSize" value="10MB"/>
    <param name="MaxBackupIndex" value="10"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d{ISO8601} %-5p [%c{1}] %m%n" />
</layout>
</appender>
```

If you want to specify an absolute file path for the log files, then you will need to update the appender configuration to use the regular log4j appender class.

In `portal-log4j-ext.xml`, make the following changes to the appender configuration:

1. Change the class to be `org.apache.log4j.RollingFileAppender`.
2. Change the value of the `File` parameter to specify the full path to the file.

For example:

```
<appender name="FILE" class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="C:\endeca-portal\logs\eid-studio.log"/>
    <param name="MaxFileSize" value="10MB"/>
    <param name="MaxBackupIndex" value="10"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d{ISO8601} %-5p [%c{1}] %m%n" />
</layout>
</appender>
```

About metrics logging

An additional file appender captures metrics logging, including all log entries from the `com.endeca.portal.instrumentation` classes.

The metrics log file, `eid-studio-metrics.log`, is in the same directory as `eid-studio.log`.

You also can view metrics data on the **Performance Metrics** page. For details on metrics logging, see *Monitoring the Performance of Queries on page 12*.

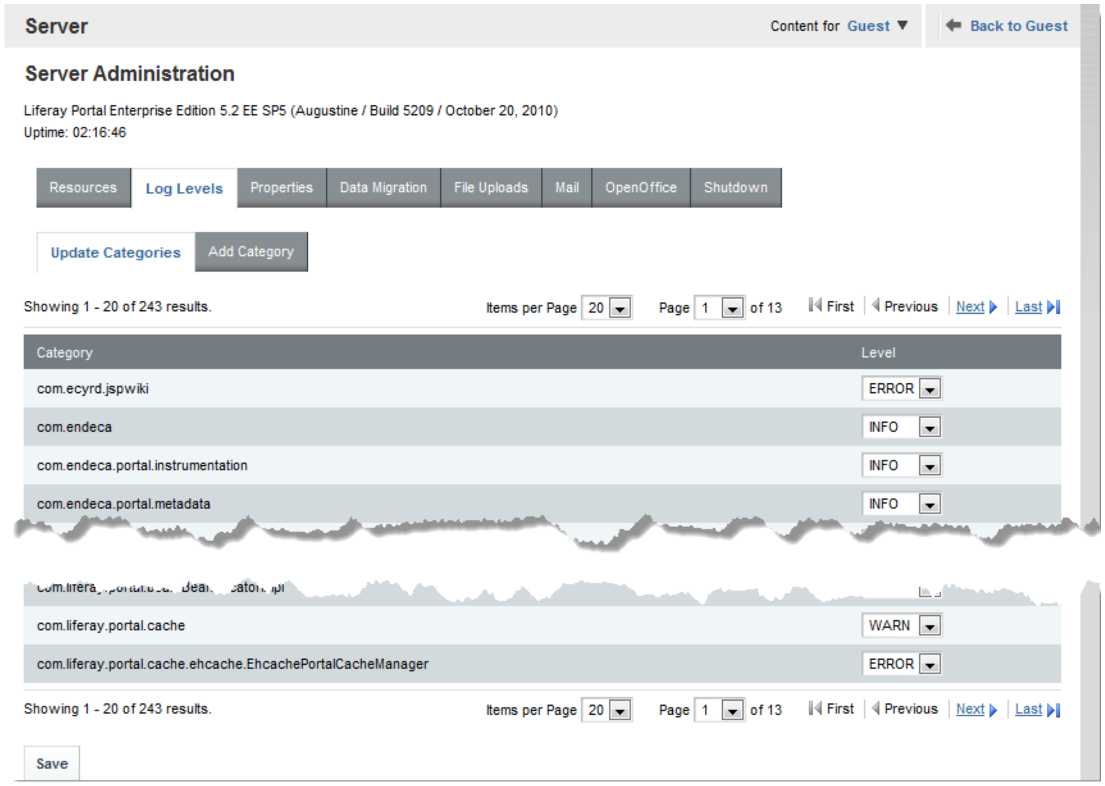
Using the Control Panel to adjust logging verbosity

For debugging purposes in a development environment, you can use the **Control Panel** to dynamically adjust logging verbosity levels for any class hierarchy.

Note: When you adjust the logging verbosity, it is updated for both `log4j` and the Java Utility Logging Implementation (JULI). Code using either of these loggers should respect this configuration.

To adjust logging verbosity from the **Control Panel**:

1. From the Studio menu, choose **Control Panel**.
2. From the **Control Panel** menu, choose **Server Administration**.
3. In the **Server Administration** page, click the **Log Levels** tab.



Liferay Portal Enterprise Edition 5.2 EE SP5 (Augustine / Build 5209 / October 20, 2010)
Uptime: 02:16:46

Showing 1 - 20 of 243 results.

Category	Level
com.ecyrd.jspwiki	ERROR
com.endeca	INFO
com.endeca.portal.instrumentation	INFO
com.endeca.portal.metadata	INFO
com.liferay.portal.cache	WARN
com.liferay.portal.cache.ehcache.EhcachePortalCacheManager	ERROR

Showing 1 - 20 of 243 results.

4. On the **Update Categories** tab, locate the class hierarchy you want to modify.
5. From the logging level drop-down list, select the logging level.

   Note: When you modify a class hierarchy, all classes that fall under that class hierarchy also are changed.

6. When you have finished adjusting log levels, click **Save**.

You also can use the **Add Category** tab to set the verbosity for a specific class or package.

How log4j.properties is used in Studio

In Studio, a `log4j.properties` configuration file is used to start up the Tomcat bundle. Otherwise, all log4j configuration comes from the XML configuration files.

Using log4j.properties when starting up the Tomcat bundle

The version of `log4j.properties` in `common/endorsed/log4j.properties.jar` is used to configure logging for the Tomcat bundle. Because `log4j` is initialized before Studio in the Tomcat bundle, this file is needed to ensure that there is some preliminary `log4j` configuration.

This version of `log4j.properties` provides minimal configuration, so that the initial messages are logged to the console in the same format as the default configuration in `portal-log4j-ext.xml`. Its settings only affect a small number of messages printed as the server is starting.

When Studio starts, it loads its XML configuration file, which overrides the settings in `log4j.properties`. Therefore administrators should not need to modify `log4j.properties`.

Other log4j.properties files

In addition to the `log4j.properties` file used for the Tomcat startup, all deployed portlets, as well as the Studio application itself, have their own `log4j.properties` files, located in `WEB-INF/classes`.

Because Studio uses XML configuration files, the settings in these files have no effect.

Chapter 3

Monitoring the Performance of Queries

You can get access to performance metrics data both from the metrics log file and from the **Performance Metrics** page. A setting in `portal-ext.properties` controls the amount of metrics data to record.

Configuring the amount of metrics data to record

To configure the metrics you want to include, you use a setting in `portal-ext.properties`. This setting applies to both the `eid-studio-metrics.log` file and the **Performance Metrics** page.

The metrics logging can include:

- Endeca Server queries by data source
- Portlet server executions by component. The server side code is written in Java.

  It handles configuration updates, configuration persistence, and Endeca Server queries. The server side code generates results to send back to the client side code.

  Server executions include portlet render, resource, and action requests.
- Portlet client executions by component. The client side code lives in the browser and is written in JavaScript.

  It issues requests to the server code, then renders the results as HTML. The client code also handles any dynamic events within the browser.

By default, only the Endeca Server queries and portlet server executions are included.

You use the `df.performanceLogging` setting in `portal-ext.properties` to configure the metrics to include. The setting is:

```
df.performanceLogging=<metrics to include>
```

Where <*metrics to include*> is a comma-separated list of the metrics to include. The available values to include in the list are:

Value	Description
`QUERY`	If this value is included, then the page includes information for Endeca Server queries.

Value	Description
`PORTLET`	If this value is included, then the page includes information on portlet server executions.
`CLIENT`	If this value is included, then the page includes information on portlet client executions.

In the default configuration, where only the Endeca Server queries and portlet server executions are included, the value is:

```
df.performanceLogging=QUERY,PORTLET
```

To include all of the available metrics, you would add the `CLIENT` option:

```
df.performanceLogging=QUERY,PORTLET,CLIENT
```

Note that for performance reasons, this configuration is not recommended.

If you make the value empty, then the log file and **Performance Metrics** page also are empty.

```
df.performanceLogging=
```

About the metrics log file

The `eid-studio-metrics.log` file contains the metrics logging information. It is located in the same directory as the `eid-studio.log` file.

The metrics log file contains the following columns:

Column Name	Description
Total duration (msec)	The total time for this entry (End time minus Start time).
Start time (msec since epoch)	The time when this entry started. For Endeca Server queries and server executions, uses the server's clock. For client executions, uses the client's clock.
End time (msec since epoch)	The time when this entry was finished. For Endeca Server queries and server executions, uses the server's clock. For client executions, uses the client's clock.
Session ID	The session ID for the client.
Page ID	If client instrumentation is enabled, the number of full page refreshes or actions the user has performed. Used to help determine how long it takes to load a complete page. Some actions that do not affect the overall state of a page, such as displaying attributes on a **Guided Navigation** component, do not increment this counter.

Column Name	Description
Gesture ID	The full count of requests to the server.
Portlet ID	This is the ID associated with an individual instance of a component. It generally includes: • The type of component • A unique identifier For example, if a page includes two **Chart** components, the ID can be used to differentiate them.
Entry Type	The type of entry. For example: • `PORTLET_RENDER` - Server execution in response to a full refresh of a component • `DISCOVERY_SERVICE_QUERY` - Endeca Server query • `CONFIG_SERVICE_QUERY` - Configuration service query • `SCONFIG_SERVICE_QUERY` - Semantic configuration service query • `LQL_PARSER_SERVICE_QUERY` - EQL parser service query • `CLIENT` - Client side JavaScript execution • `PORTLET_RESOURCE` - Server side request for resources • `PORTLET_ACTION` - Server side request for an action
Miscellaneous	A URL encoded JSON object containing miscellaneous information about the entry.

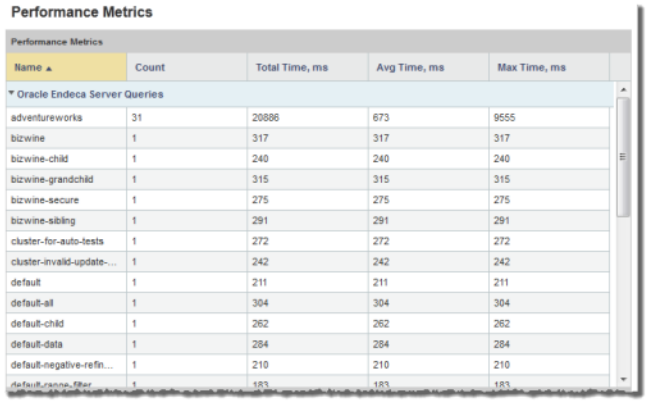
About the Performance Metrics page

The **Performance Metrics** page on the **Control Panel** displays information about component and Endeca Server query performance.

It uses the same logging data recorded in `eid-studio-metrics.log`.

However, unlike the log file, the **Performance Metrics** page uses data stored in memory. Restarting Studio clears the **Performance Metrics** data.

For each type of included metric, the table at the top of the page contains a collapsible section.



Performance Metrics

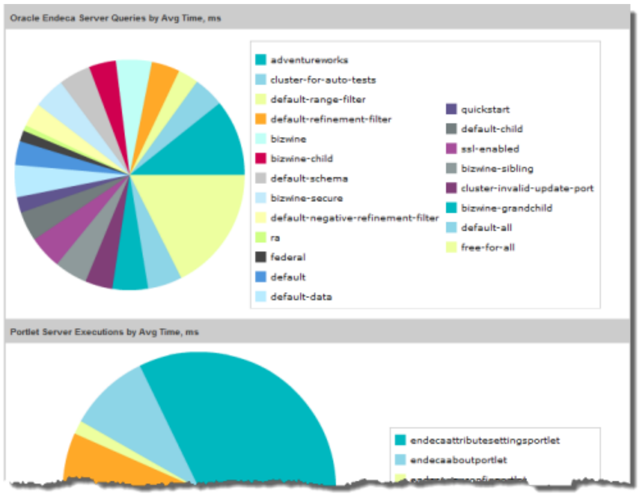
Performance Metrics

Name ▲	Count	Total Time, ms	Avg Time, ms	Max Time, ms
▾ Oracle Endeca Server Queries				
adventureworks	31	20886	673	9555
bizwine	1	317	317	317
bizwine-child	1	240	240	240
bizwine-grandchild	1	315	315	315
bizwine-secure	1	275	275	275
bizwine-sibling	1	291	291	291
cluster-for-auto-tests	1	272	272	272
cluster-invalid-update-...	1	242	242	242
default	1	211	211	211
default-all	1	304	304	304
default-child	1	262	262	262
default-data	1	284	284	284
default-negative-refin...	1	210	210	210
default-range-filter	1	183	183	183

For each data source or component, the table tracks:

- Total number of queries or executions
- Total execution time
- Average execution time
- Maximum execution time

For each type of included metric, there is also a pie chart summarizing the average query or execution time per data source or component.



Note: Endeca Server query performance does not correlate directly to a Studio application page, as a single Studio page often uses multiple Endeca Server queries.

Chapter 4

Determining and Configuring the Locale Used in Studio

The Studio UI can be displayed in different locales.

About locales and their effect on the UI

The locale determines the language in which to display the Studio UI. It can also affect the format of displayed data values.

Studio is configured with a default locale as well as a list of available locales.

Each user account also is configured with a preferred locale, and the Studio menu includes an option for users to select the locale to use.

How Studio determines the locale to use

When users enter Studio, it needs to determine the locale to use to display the UI.

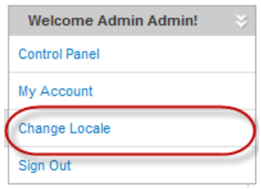
Locations where the locale may be set

The locale is set in different locations in Studio.

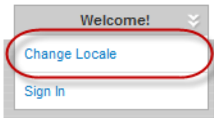
Studio can get the locale from the following locations:

- Studio cookie
- Browser locale
- Studio default locale
- User preferred locale, stored as part of the user account

- Locale selected using the **Change locale** option in the Studio menu.



The menu option also is available to users who have not yet logged in.



- Locale provided as part of a deep linking URL into Studio. For example:

```
http://localhost:8080/web/myapp/my-
page?doAsUserLanguageId=zh_CN&deeplink=[{"default":{queryFunctions:[{"class":"R
efinementFilter","attributeValue":"1997", "attributeKey":"Vintage"}]}}]
```

Scenarios for selecting the locale

The locale used depends upon the type of user, the Studio configuration, and how the user entered Studio.

For the scenarios listed below. Studio determines the locale as follows:

Scenario	How the locale is determined
A new Studio user is created	The locale for a new user is initially set to **Use Browser Locale**, which indicates to use the current browser locale. This value can be changed to a specific locale. If the user is configured with a specific locale, then that locale is used for that user unless they explicitly select a different locale or they enter Studio with a URL that includes a supported locale.
A non-logged-in user navigates to Studio	For a non-logged-in user, Studio first tries to use the locale from the cookie. If there is no cookie, or the cookie is invalid, then Studio tries to use the browser locale. If the current browser locale is not one of the supported Studio locales, then Studio uses its configured default locale.

Scenario	How the locale is determined
A registered user logs in to Studio	When a user logs in, Studio first checks the locale configured for their user account. • If the user's locale is set to **Use Browser Locale**, then Studio tries to use the locale from the cookie. If there is no cookie, or if the cookie is invalid, then Studio tries to use the browser locale. If the current browser locale is not a supported Studio locale, then Studio uses its configured default locale. • If the user account is configured with a locale value other than **Use Browser Locale**, then Studio uses that locale, and also updates the cookie with that locale.
A non-logged-in user uses the Studio menu option to select a different locale	When a non-logged-in user selects a locale, Studio updates the cookie with the new locale. Note that this locale change is only applied locally. It is not applied to all non-logged-in users.
A logged-in user uses the Studio menu option to select a different locale	When a logged-in user selects a locale, Studio updates both the user's account and the cookie with the selected locale.
A non-logged-in user navigates to Studio using a URL that includes a locale	If the locale from the URL is supported by Studio, then Studio uses that locale and also updates the cookie with that locale. If the URL locale is not a supported Studio locale, then Studio tries to use the locale from the cookie. If there is no cookie, or if the cookie is invalid, then Studio tries to use the browser locale. If the current browser locale is not one of the supported Studio locales, then Studio uses its configured default locale.

Scenario	How the locale is determined
A logged-in user navigates to Studio using a URL that includes a locale	If the URL locale is supported by Studio, then Studio uses that locale. Studio updates both the cookie and the user's account to reflect that URL. If the URL locale is not a supported Studio locale, then Studio gets the locale configured for the user's account. • If the user's locale is set to **Use Browser Locale**, then Studio tries to use the locale from the cookie. If there is no cookie, or if the cookie is invalid, then Studio tries to use the browser locale. If the current browser locale is not a supported Studio locale, then Studio uses its configured default locale. • If the user's locale is a value other than **Use Browser Locale**, then Studio uses that locale and also updates the cookie with that locale.

Setting the available locales for Studio

Studio is configured with a list of available locales. This list is used to populate the drop-down list for configuring the Studio default locale, user default locale, and the available locales displayed for the **Change locale** option.

You can customize the setting to constrain the list. Supported locales are specified in `portal.properties`.

```
locales=de_DE, en_US, es_ES, fr_FR, it_IT, ja_JP, ko_KR, pt_PT, zh_CN, zh_TW
```

To reduce this list:

1. Copy this parameter into `portal-ext.properties`.
2. Update the list to remove the locales that you do not want to be available.

   For example, to only support English, French, and Japanese, you would update it to:

   ```
   locales=en_US, fr_FR, ja_JP
   ```

Setting the default locale used by Studio

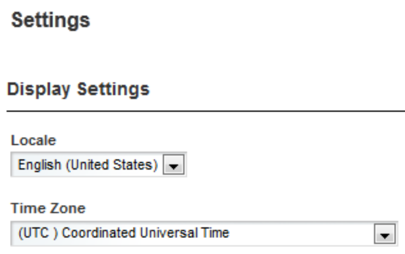
Studio is configured with a default locale, which you can update from the Control Panel.

Note that if you have a clustered implementation, make sure to configure the same locale for all of the instances in the cluster.

To configure the default locale for an instance of Studio:

1. From the **Studio** menu, select **Control Panel**.
2. On the **Control Panel** menu, in the **Portal** section, click **Settings**.
3. On the **Portal** page, in the menu on the right, click **Display Settings**.

4. On the **Display Settings** page, from the **Locale** drop-down list, select the default locale for studio.



Settings

Display Settings

Locale: English (United States)

Time Zone: (UTC) Coordinated Universal Time

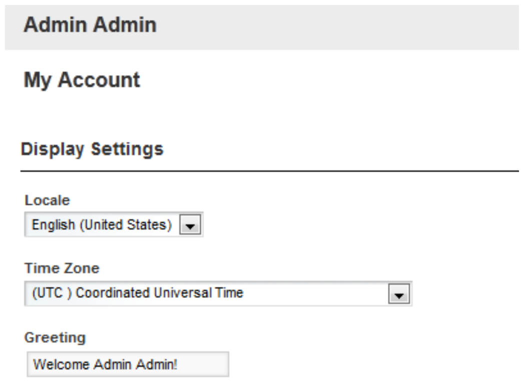
5. Click **Save**.

Configuring the preferred locale for a Studio user

Each user account is configured with a preferred locale. The default value for new users is **Use Browser Locale**, which indicates to use the current browser locale.

To configure the default locale for a user:

1. To display the setting for your own account:
   (a) From the Studio menu, select **My Account**.
   (b) On the **My Account** page, in the menu on the right, click **Display Settings**.



Admin Admin

My Account

Display Settings

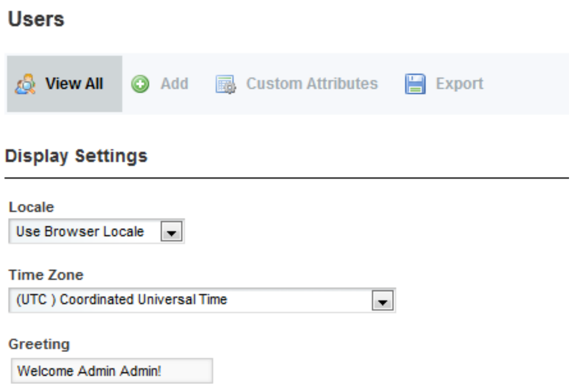
Locale: English (United States)

Time Zone: (UTC) Coordinated Universal Time

Greeting: Welcome Admin Admin!

2. To display the setting for another user:
   (a) From the Studio menu, select **Control Panel**.
   (b) In the **Control Panel** menu, under **Portal**, click **Users**.
   (c) On the **Users** page, click the **Actions** button for the user you want to edit.
   (d) From the **Actions** menu, select **Edit**.

(e) On the user edit page, in the menu on the right, click **Display Settings**.



Locale
Use Browser Locale

Time Zone
(UTC) Coordinated Universal Time

Greeting
Welcome Admin Admin!

3. From the **Locale** drop-down list, select the preferred locale for the user.
4. Click **Save**.

Including the locale in a URL

To include the locale in the URL, add the locale as a value of the `doAsUserLanguageId` parameter.

For example, to include the locale in a deep linking URL:

```
http://localhost:8080/web/myapp/my-
page?doAsUserLanguageId=zh_CN&deeplink=[{"default":{queryFunctions:[{"class":"Refi
nementFilter","attributeValue":"1997", "attributeKey":"Vintage"}]}}]
```

Part II

Controlling Access to Studio

Chapter 5

About Managing Users in Studio

You can create users directly in Studio, or connect to an LDAP system.

About user roles

About the default user

About user roles

In Studio, each user is assigned a user role. The user's role controls the functions that the user has access to.

The basic user roles are:

Role	Description
Power User	For a new user, the default role is Power User. These users can: • View Studio applications, based on the application and page type and their application membership • Create Studio applications • Configure and manage applications for which they are an administrator • Edit their account information They do not have access to **Control Panel** functions.
Administrator	Administrators have full access to Studio and Studio applications. They can: • View all Studio applications • Create Studio applications • Configure and manage all Studio applications • Use all of the **Control Panel** functions

For information on application access and assigning application roles to users, see the *Studio User's Guide*.

About the default user

When you first install Studio, a default user is created.

The default user is an administrator and has full privileges to:

- View all Studio applications and pages, including private applications and pages
- Create Studio applications
- Configure and manage all Studio applications, including private applications
- View and use all of the **Control Panel** functions

To log in as the default user for the very first time, use the following user name and password:

Field	Value to Enter
Email address:	admin@oracle.com
Password:	Welcome123

You are immediately prompted to change the password. The new password must contain:

- At least 6 characters
- At least one non-alphabetic character

Chapter 6

Creating and Editing Users in Studio

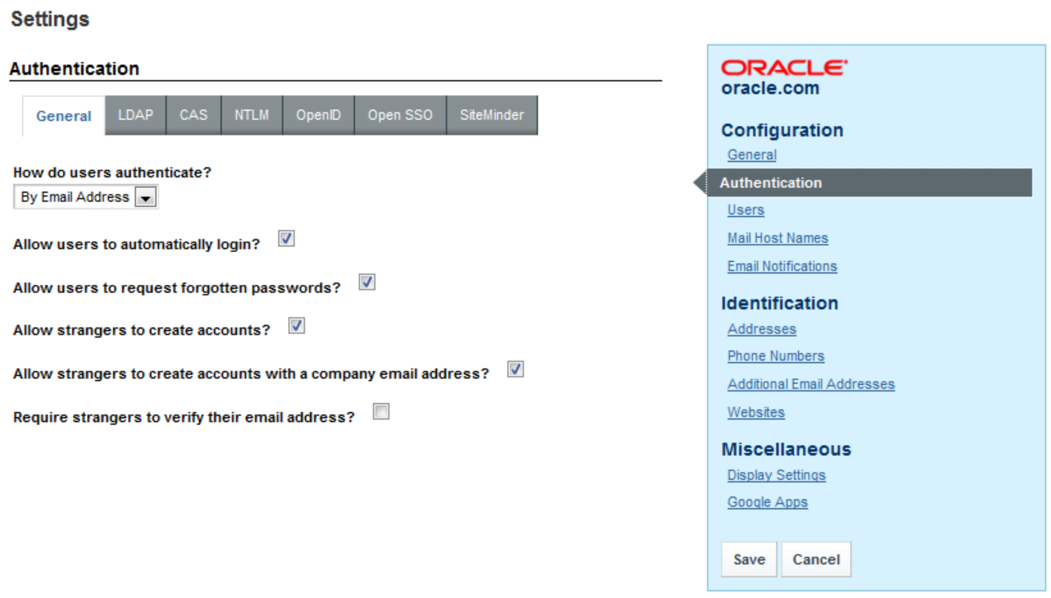
The **Users** page provides options for creating and editing Studio users.

Configuring the type of user name for Studio

Each Studio user has both an email address and a screen name. By default, users log in to Studio using their email address.

To change the configuration so that users log in with their screen name:

1. From the Studio menu, select **Control Panel**.
2. On the **Control Panel** menu, click **Settings**.
3. In the **Settings** page menu to the right, click **Authentication**.
4. On the **General** tab, from the **How do users authenticate?** drop-down list, select the name used to log in.



How do users authenticate? By Email Address

Allow users to automatically login? ✓

Allow users to request forgotten passwords? ✓

Allow strangers to create accounts? ✓

Allow strangers to create accounts with a company email address? ✓

Require strangers to verify their email address? ☐

5. Click **Save**.

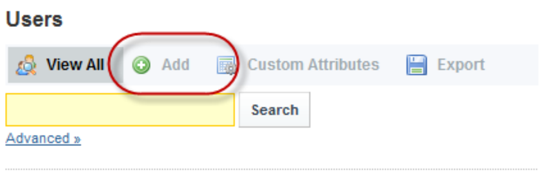
Creating a new user

Even if you are importing users from LDAP, you may still want to create a few users directly in Studio.

For example, for a small development instance, you may just need a few users to develop and test pages. Or if your LDAP users for a production site are all end users, you may need a separate user account for administering and editing the site.

To add a new user:

1. From the Studio menu, select **Control Panel**.
2. On the **Control Panel** menu, click **Users**.
3. On the **Users** page, click **Add**.



4. On the **Details** page, to provide the minimum required information:
   (a) In the **Screen Name** field, type the screen name for the user.

   The screen name must be unique, and cannot match the screen name of any current active or inactive user.

   (b) In the **Email Address** field, type the user's email address.
   (c) In the **First Name** field, type the user's first name.
   (d) In the **Last Name** field, type the user's last name.
5. Click **Save**.

   The new user is added, and the configuration menu is updated to add the rest of the options.
6. To create the password for the user:
   (a) On the user page, on the configuration menu to the right, click **Password**.
   (b) On the **Password** page, enter the password to assign to the new user.
   (c) Click **Save**.
7. To add the user to an application:
   (a) On the user page, from the list to the right, click **Applications**.
   (b) Click the **Select** link.
   (c) On the applications list, click the application to add the user to.
8. To assign a role to the user:
   (a) On the user page, from the list to the right, click **Roles**.
   (b) Select the roles for the user.
   (c) Click **Save**.

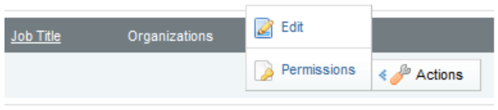
9. For the Power User role, in order for the user to be able to manage pages, they must be a Application Administrator or Application Owner. On the **Roles** page for the user:
   (a) Under **Application Roles**, click the **Select** link.
   (b) On the application roles list, click the role you want to assign to the user.
   (c) Click **Save**.

Editing a Studio user

The **Users** page also allows you to edit a user's account.

From the **Users** page, to edit a user:

1. Click the **Actions** button next to the user.
2. In the **Actions** menu, click **Edit**.



3. After making your changes, click **Save**.

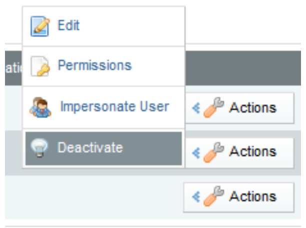
Deactivating, reactivating, and deleting Studio users

From the **Users** page of the **Control Panel**, you can make an active user inactive. You can also reactivate or delete inactive users.

Note that you cannot make your own user account inactive, and you cannot delete an active user.

From the **Users** page, to change the status of a user account:

1. To make an existing user inactive:
   (a) In the users list, click the **Actions** button for the user you want to deactivate.



   (b) From the **Actions** menu, select **Deactivate**.

   Studio prompts you to confirm that you want to deactivate the user.

   The user is then removed from the list of active users.

   Note that inactive users are not removed from Studio.

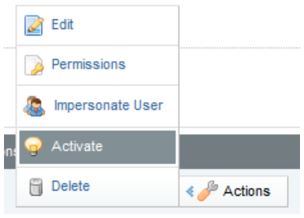
2. To reactivate or delete an inactive user:
   (a) Click the **Advanced** link below the user search field.

   Studio displays additional user search fields.

   (b) From the **Active** drop-down list, select **No**.
   (c) Click **Search**.

   The users list displays only the inactive users.

   (d) Click the **Actions** button for the use you want to re-activate.



   (e) To reactivate the user, from the **Actions** menu, select **Activate**.
   (f) To delete the user, from the **Actions** menu, select **Delete**.

Chapter 7

Integrating with an LDAP System to Manage Users

If you have an LDAP system, you can allow users to use those credentials to log in to Studio.

About using LDAP

LDAP (Lightweight Directory Access Protocol) allows you to have users connect to your Studio application using their existing LDAP user accounts, rather than creating separate user accounts from within Studio. LDAP is also used when integrating with a single sign-on (SSO) system.

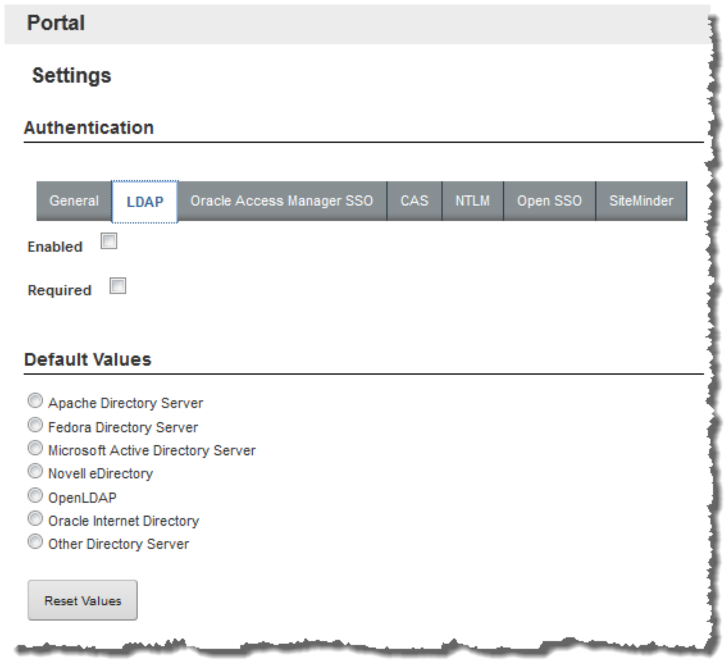
Configuring the LDAP settings and server

You configure the LDAP connection from the **Control Panel**. The settings include whether LDAP is enabled and required for authentication, the connection to the LDAP server, and whether to import or export users to or from the LDAP directory.

To display the LDAP configuration page and configure the basic settings:

1. From the Studio menu, click **Control Panel**.
2. On the **Control Panel** menu, click **Settings**.
3. In the **Settings** page menu to the right, click **Authentication**.

4. Click the **LDAP** tab.

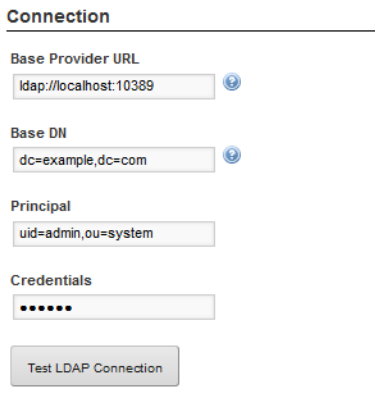


5. On the **LDAP** tab:
6. To enable LDAP authentication, check the **Enabled** checkbox.
7. To only allow users to log in using an LDAP account, check the **Required** checkbox.

   If this box is checked, then any users that you create manually in Studio cannot log in.

   To make sure that users you create manually can log in, make sure that this box is not checked.
8. To populate the LDAP server configuration fields with default values based on a specific type of server:
   (a) Under **Default Values**, click the radio button for the type of server you are using.
   (b) Click **Reset Values**.

9. The **Connection** settings cover the basic connection to LDAP:



Base Provider URL: ldap://localhost:10389

Base DN: dc=example,dc=com

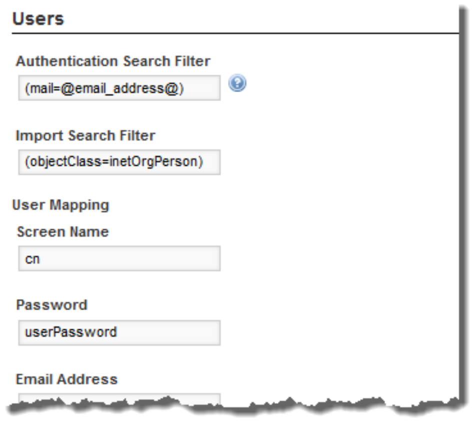
Principal: uid=admin,ou=system

Credentials: ••••••

Field	Description
Base Provider URL	The location of your LDAP server. Make sure that the machine on which Studio is installed can communicate with the LDAP server. If there is a firewall between the two systems, make sure that the appropriate ports are opened.
Base DN	The Base Distinguished Name for your LDAP directory. For a commercial organization, it may look something like: `dc=companynamehere,dc=com`
Principal	The user name of the administrator account for your LDAP system. This ID is used to synchronize user accounts to and from LDAP.
Credentials	The password for the administrative user.

After providing the connection information, to test the connection to the LDAP server, click the **Test LDAP Connection** button.

10. The **Users** section contains settings for finding users in your LDAP directory. The first couple of settings are filters for finding and identifying users.



Users

Authentication Search Filter
(mail=@email_address@)

Import Search Filter
(objectClass=inetOrgPerson)

User Mapping

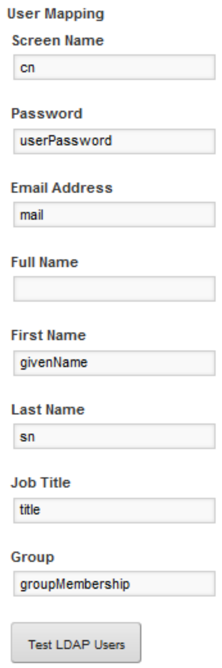
Screen Name
cn

Password
userPassword

Email Address

Field	Description
Authentication Search Filter	Determines the search criteria for user logins. By default, users log in using their email address. If you have changed this setting, you must modify the search filter here. For example, if you changed the authentication method to use the screen name, you would modify the search filter so that it can match the entered login name: `(cn=@screen_name@)`
Import Search Filter	Depending on the LDAP server, there are different ways to identify the user. The default setting (`objectClass=inetOrgPerson`) usually is fine, but to search for only a subset of users or for users that have different object classes, you can change this.

11. Under **User Mapping**, map your LDAP attributes to the Studio user fields:

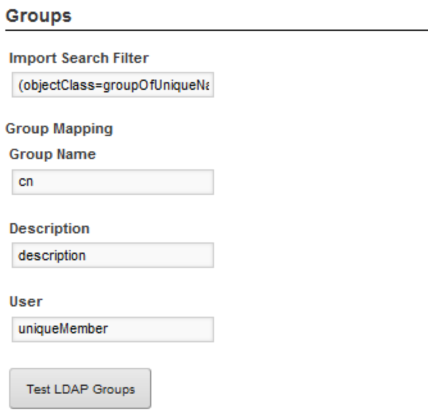


User Mapping

Screen Name	cn
Password	userPassword
Email Address	mail
Full Name	
First Name	givenName
Last Name	sn
Job Title	title
Group	groupMembership

After setting up the attribute mappings, to test the mappings, click **Test LDAP Users**.

12. Under **Groups**, map your LDAP groups.



Groups

Import Search Filter	(objectClass=groupOfUniqueNa

Group Mapping

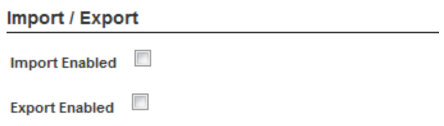
Group Name	cn
Description	description
User	uniqueMember

In the **Import Search Filter** field, type the filter for finding LDAP groups, then map the following fields:

- Group Name
- Description
- User

To test the group mappings, click **Test LDAP Groups**. The system displays a list of the groups returned by your search filter.

13. The **Import/Export** section is used to configure importing and exporting of LDAP user data:


Import / Export

Import Enabled ☐

Export Enabled ☐

   (a) If the **Import Enabled** checkbox is checked, then when you start Studio, it can import all of your LDAP groups and users.

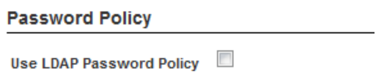
   If the box is not checked, then Studio synchronizes each user as they log in.

   It is recommended that you leave this box unchecked.

   (b) If the **Export Enabled** checkbox is checked, then any changes to the user in Studio are exported to the LDAP system.

   It is recommended that you leave this box unchecked.

14. To use the password policy from your LDAP system, instead of the Studio password policy, check the **Use LDAP Password Policy** checkbox.


Password Policy

Use LDAP Password Policy ☐

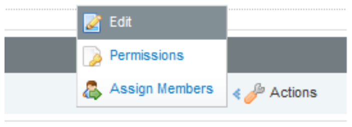
15. To save the LDAP configuration, click **Save**.

Configuring the Studio password policy when using LDAP

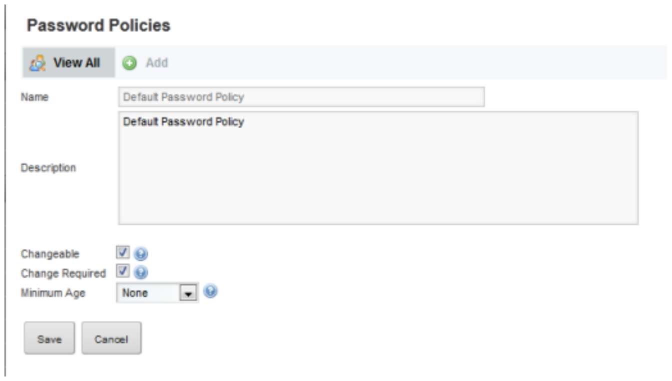
When you are using LDAP, it is likely that you want user passwords to be managed outside of Studio. So if you are not using the LDAP password policy, then you may want to update the password policy to prevent users from changing their password in Studio.

To update the password policy:

1. From the Studio menu, select **Control Panel**.
2. In the **Control Panel** menu, click **Password Policies**.
3. On the **Password Policies** page, click the **Actions** button, then click **Edit**.



4. On the **Password Policies** page:



Name: Default Password Policy
Description: Default Password Policy
Changeable: ✓
Change Required: ✓
Minimum Age: None

   (a) To prevent users from being able to change passwords from within Studio, uncheck the **Changeable** checkbox.
   (b) To prevent users from being prompted to change their password the first time they log in to Studio, uncheck the **Change Required** checkbox.

5. To save the changes, click **Save**.

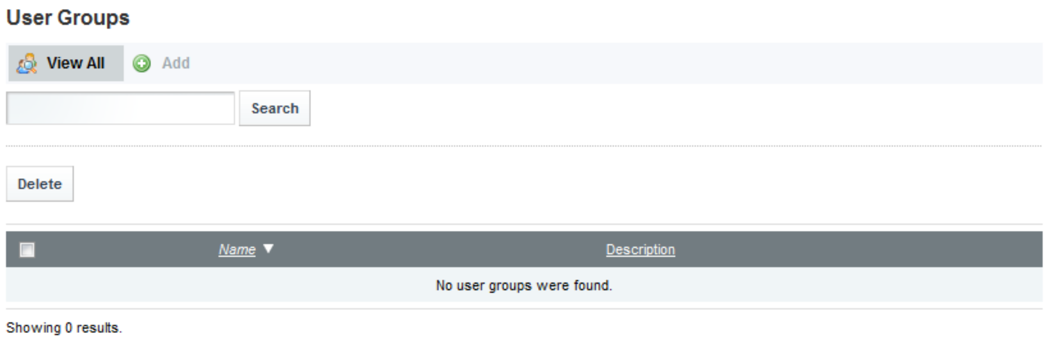
Assigning roles based on LDAP user groups

For LDAP integration, it is recommended that you assign roles based on your LDAP groups.

To ensure that users have the correct roles as soon as they log in, you create groups in Studio that have the same name as your LDAP groups, but in lowercase, then assign the correct roles to each group.

To create a group and then assign a role to that group:

1. From the Studio menu, select **Control Panel**.
2. On the **Control Panel**, click **User Groups**.

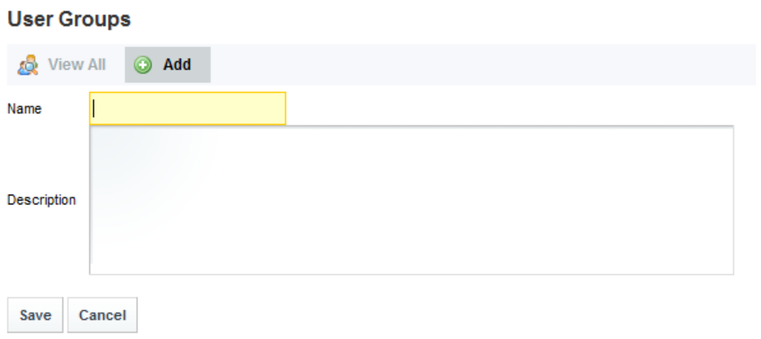


	Name ▼	Description
No user groups were found.		

Showing 0 results.

3. On the **User Groups** page, to add a new group:
   (a) Click the **Add** button.

   The new group page is displayed.



   (b) On the new group page, in the **Name** field, type the name of the group.

   Make sure the name is the lowercase version of the name of a group from your LDAP system. For example, if the LDAP group is called SystemUsers, then the user group name would be systemusers.

   (c) In the **Description** field, type a description of the group.
   (d) Click **Save**.

   The group is added to the **User Groups** list.

4. To assign the group to a role:
   (a) In the **Control Panel** menu, click the **Roles** option.
   (b) On the **Roles** page, for the role you want to assign the group to, click the **Actions** button.
   (c) In the menu, click **Assign Members**.
   (d) Click the **User Groups** tab.
   (e) To display the list of available groups to assign to the role, click the **Available** tab.
   (f) Check the checkbox next to the group, then click the **Update Associations** button.

   The group is added to the **Current** tab as a group assigned that user role.

Chapter 8

Setting up Single Sign-On (SSO) for Studio

Studio supports integrating with an SSO system.

About single sign-on and Studio

Integrating Studio with single sign-on (SSO) allows your users to be logged in to Studio automatically once they are logged in to your system.

Note that once Studio is integrated with SSO, you cannot create and edit users from within Studio. All users get access to Studio using their SSO credentials. This means that you will no longer be able to use the default administrative user provided with Studio. You will need to make sure that there is at least one SSO user with an Administrator user role for Studio.

Studio does provide multiple options for SSO integration, including integrating with OpenSSO or SiteMinder.

However, the recommended method for integrating with SSO, and the method we have tested and are documenting here, is to use Oracle Access Manager, with an Oracle HTTP Server in front of the Studio application server.

The information in this guide focuses on the details and configuration that are specific to the Studio integration. For general information on installing Oracle Access Manager and Oracle HTTP Server, see the associated documentation for those products.

Overview of the process for configuring SSO with Oracle Access Manager

Here is an overview of the steps for using Oracle Access Manager to implement SSO in Studio.

1. Install Oracle Access Manager 11g, if you haven't already. See the Oracle Access Manager documentation for details.
2. Install Oracle HTTP Server (OHS) 11g. See the Oracle HTTP Server documentation for details.

3. Install OHS Webgate 11g. See the Webgate documentation for details.
4. Create an instance of OHS, and confirm that it is up and running. See the OHS documentation for details.
5. Configure the reverse proxy module for the Studio application server in Oracle HTTP Server. See *Configuring the reverse proxy module in OHS on page 38*.
6. Install the Webgate module into the Oracle HTTP Server. See *Registering the Webgate with the Oracle Access Manager server on page 39*.
7. In Studio, configure the LDAP connection for your SSO implementation. See *Configuring the LDAP connection for SSO on page 42*.
8. In Studio, configure the Oracle Access Manager SSO settings. See *Configuring the Oracle Access Manager SSO settings on page 43*.
9. Configure Studio's web server settings to use the OHS server. See *Completing and testing the SSO integration on page 44*.
10. Disable direct access to the Studio application server, to ensure that all traffic to Studio is routed through OHS.

Configuring the reverse proxy module in OHS

You must configure your OHS instance to pass traffic back to Studio as a reverse proxy. Use the appropriate instructions for your Studio application server (Tomcat or WebLogic).

Reverse proxy configuration for Tomcat

Reverse proxy configuration for WebLogic

Reverse proxy configuration for Tomcat

For Tomcat, the reverse proxy configuration consists of a `tomcat-proxy.conf` file that contains the reverse proxy settings.

To configure the reverse proxy for Tomcat:

1. Find the `config` directory for your OHS instance (`MW_HOME/Oracle_WT1/instances/INST_DIR/config/OHS/ohs1/`).
2. 2. Within the `moduleconf/` subdirectory, create a file called `tomcat-proxy.conf` with the following content:

```
ProxyPass /oberr.cgi !
ProxyPass /oam_logout_success http://hostName:tomcatPort/c/portal/logout
ProxyPassReverse /oam_logout_success http://hostName:tomcatPort/c/portal/logout
ProxyPass / http://hostName:tomcatPort/
ProxyPassReverse / http://hostName:tomcatPort/
```

   Where:

   - *`hostName`* is the host name for the Studio application.
   - *`tomcatPort`* is the port number used by Tomcat for Studio.

The `/oam_logout_success` proxy rule redirects this request (after the Webgate handles it) to Studio's logout URL. With this configuration, when users sign out of SSO from another application, it is reflected in Studio.

3. Restart the OHS instance using the `opmnctl` command, which is run on the command line from `MW_HOME/Oracle_WT1/instances/INST_DIR`.

   ```
   bin/opmnctl restartproc
   ```

4. Test the OHS URL in your browser.

   You should be forwarded to Studio.

Reverse proxy configuration for WebLogic

For WebLogic, you need to update the file `mod_wls_ohs.conf` to add the logout configuration for SSO.

Here is an example of the file with the `/oam_logout_success` section added:

```
LoadModule weblogic_module   "${ORACLE_HOME}/ohs/modules/mod_wl_ohs.so"
<IfModule weblogic_module>
      WebLogicHost hostName
      WebLogicPort portNumber
</IfModule>

<Location /oam_logout_success>
      PathTrim /oam_logout_success
      PathPrepend /c/portal
      DefaultFileName logout
      SetHandler weblogic-handler
</Location>

<Location />
      SetHandler weblogic-handler
</Location>
```

The `/oam_logout_success Location` configuration is special for Studio. It redirects the default Webgate Logout Callback URL (`/oam_logout_success`) to an application tier logout within Studio. With this configuration, when users sign out of SSO from another application, it is reflected in Studio.

Registering the Webgate with the Oracle Access Manager server

After you have installed the OHS Webgate, you use the remote registration (RREG) tool to register the OHS Webgate with the OAM server.

To complete the registration:

1. Obtain the RREG tarball (`rreg.tar.gz`) from the Oracle Access Manager server and extract it to the OHS server.
2. Modify the script `rreg/bin/oamreg.bat` or `oamreg.sh`.

   Correct the `OAM_REG_HOME` and `JAVA_HOME` environment variables.

   `OAM_REG_HOME` should point to the extracted `rreg` directory created in the previous step.

   You may not need to change `JAVA_HOME` if it's already set in your environment.

3. In the `input` directory, create an input file for the RREG tool. The file can include the list of resources secured by this Webgate.

   You can omit this list if the application domain already exists.

   Here is an example of an input file where the resources have not been set up for the application domain and host in Oracle Access Manager:

```
<?xml version="1.0" encoding="UTF-8"?>

<OAM11GRegRequest>

<serverAddress>http://oamserver.us.mycompany.com:7001</serverAddress>
<hostIdentifier>myserver-1234</hostIdentifier>
<agentName>myserver-1234-webgate</agentName>
<applicationDomain>Information Discovery Studio</applicationDomain>
<protectedResourcesList>
  <resource>/</resource>
  <resource>/.../*</resource>
</protectedResourcesList>
<publicResourcesList>
  <resource>/public/index.html</resource>
</publicResourcesList>
<excludedResourcesList>
  <resource>/excluded/index.html</resource>
</excludedResourcesList>

</OAM11GRegRequest>
```

   In this example, the resources have already been set up in Oracle Access Manager:

```
<?xml version="1.0" encoding="UTF-8"?>

<OAM11GRegRequest>

<serverAddress>http://oamserver.us.mycompany.com:7001</serverAddress>
<hostIdentifier>myserver-1234</hostIdentifier>
<agentName>myserver-1234-webgate</agentName>
<applicationDomain>Information Discovery Studio</applicationDomain>

</OAM11GRegRequest>
```

   In the input file, the parameter values are:

Parameter Name	**Description**
`serverAddress`	The full address (`http://`*`host`*`:`*`port`*) of the Oracle Access Manager administrative server. The port is usually 7001.
`hostIdentifier`	The host identifier string for your host. If you already created a host identifier in the Oracle Access Manager console, use its name here.
`agentName`	A unique name for the new Webgate agent. Make sure it doesn't conflict with any existing agents in the application domain.

Parameter Name	Description
`applicationDomain`	A new or existing application domain to add this agent into. Each application domain may have multiple agents. An application domain associates multiple agents with the same authentication and authorization policies.

4. Run the tool:

   ```
   ./bin/oamreg.sh inband input/inputFileName
   ```

   or

   ```
   .\bin\oamreg.bat inband input\inputFileName
   ```

   For example:

   ```
   bin\oamreg.bat inband
                input\my-webgate-input.xml
   ```

   When the process is complete, you'll see the following message:

   ```
   Inband registration process completed successfully! Output artifacts are created in the
   output folder.
   ```

5. Copy the generated output files from the `output` directory to the OHS instance `config` directory (under `webgate/config/`).
6. Restart the OHS instance.
7. Test your application URL via OHS.

   It should forward you to the SSO login form.

   Check the OAM console to confirm that the Webgate is installed and has the correct settings.

Testing the OHS URL

Before continuing to the Studio configuration, you need to test that the OHS URL redirects correctly to Studio.

To test the OHS URL, use it to browse to Studio.

You should be prompted to authenticate using your SSO credentials.

Because you have not yet configured the Oracle Access Manager SSO integration in Studio, after you complete the authentication, the Studio login page is displayed.

Log in to Studio using an administrator account.

Configuring Studio to integrate with SSO via Oracle Access Manager

In Studio, you configure the LDAP connection and Oracle Access Manager connection settings.

Configuring the LDAP connection for SSO

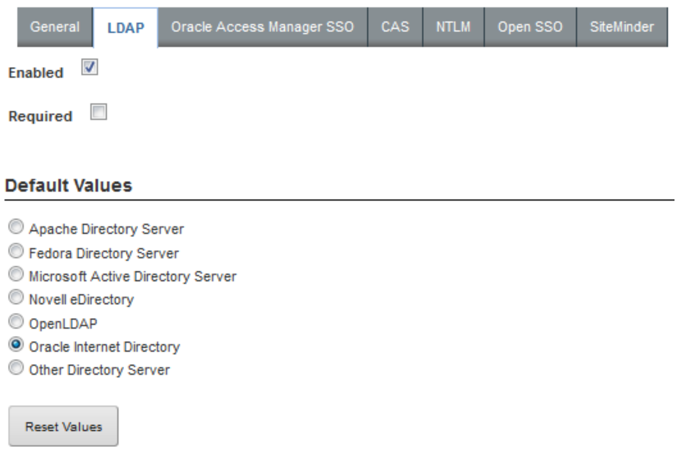
Configuring the Oracle Access Manager SSO settings

Configuring the LDAP connection for SSO

The SSO implementation uses LDAP to retrieve and maintain the user information. For the Oracle Access Manager SSO, you configure Studio to use Oracle Internet Directory for LDAP.

In Studio, to configure the LDAP connection for SSO:

1. From the Studio menu, select **Control Panel**.
2. In the **Control Panel** menu, under **Portal**, click **Settings**.
3. In the **Settings** page menu to the right, click **Authentication**.
4. On the **Authentication** page, click the **LDAP** tab.
5. Check the **Enabled** checkbox. Do not check the **Required** checkbox.
6. Under **Default values**, click the **Oracle Internet Directory** radio button, then click **Reset Values**.



Enabled [✓]

Required [ ]

Default Values

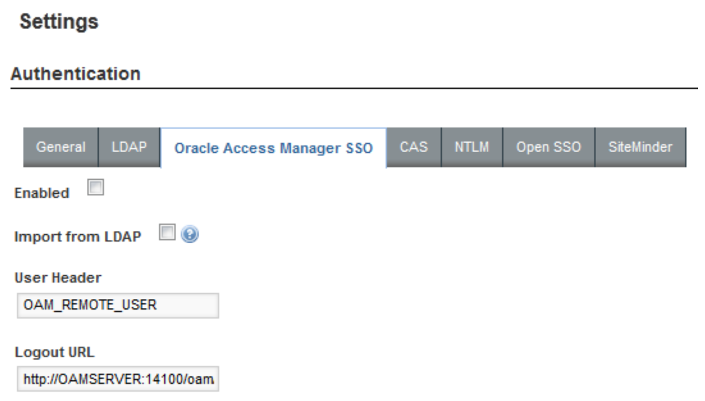
- Apache Directory Server
- Fedora Directory Server
- Microsoft Active Directory Server
- Novell eDirectory
- OpenLDAP
- ● Oracle Internet Directory
- Other Directory Server

7. Configure the LDAP connection, users, and groups as described in *Configuring the LDAP settings and server on page 29*.
8. To save the LDAP connection information, click **Save**.
9. Configure the application roles for your user groups as described in *Assigning roles based on LDAP user groups on page 35*.
10. Make sure that the password policy is configured to not require users to change their password. See *Configuring the Studio password policy when using LDAP on page 34*.

Configuring the Oracle Access Manager SSO settings

After you configure the LDAP connection for your SSO integration, you configure the Oracle Access Manager SSO settings.

The settings are on the **Oracle Access Manager SSO** tab on the **Authentication** page.



Enabled ☐

Import from LDAP ☐

User Header
OAM_REMOTE_USER

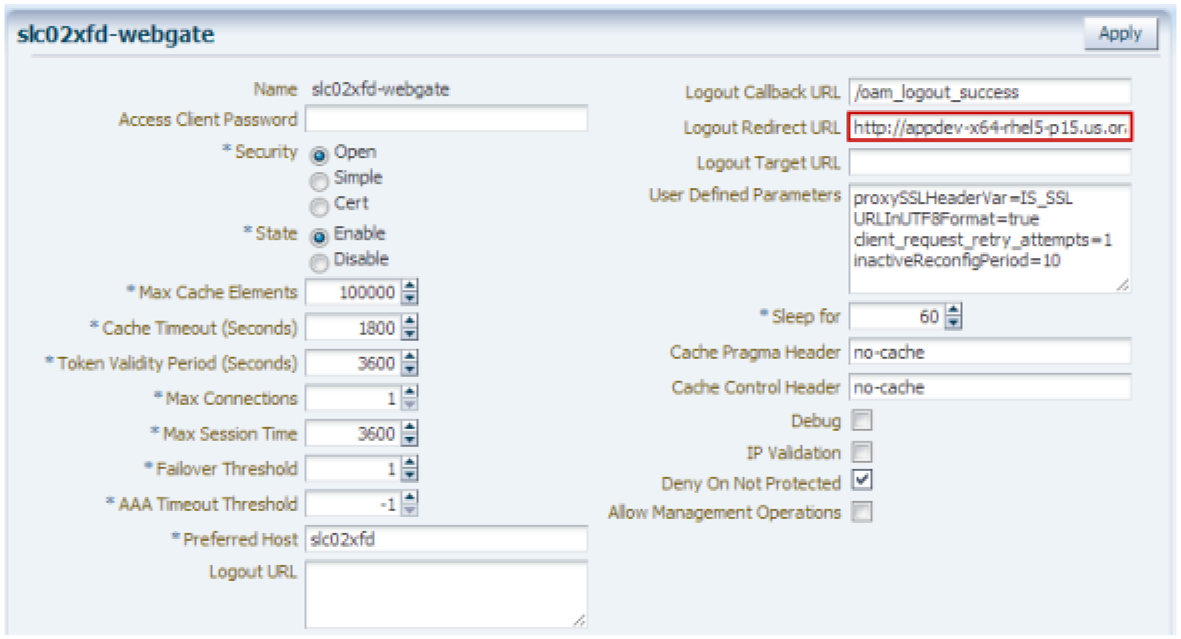
Logout URL
http://OAMSERVER:14100/oam

To configure the SSO settings:

1. From the Studio menu, select **Control Panel**.
2. In the **Control Panel** menu, under **Portal**, click **Settings**.
3. In the **Settings** page menu to the right, click **Authentication**.
4. On the **Authentication** page, click the **Oracle Access Manager SSO** tab.
5. Check the **Enabled** checkbox.
6. Check the **Import from LDAP** checkbox.
7. Leave the default user header `OAM_REMOTE_USER`.

8. In the **Logout URL** field, provide the URL to navigate to when users log out.

   Make sure it is the same logout redirect URL you have configured for the Webgate:



slc02xfd-webgate

Name	slc02xfd-webgate
Access Client Password	
* Security	[● Open] Simple Cert
* State	[● Enable] Disable
* Max Cache Elements	100000
* Cache Timeout (Seconds)	1800
* Token Validity Period (Seconds)	3600
* Max Connections	1
* Max Session Time	3600
* Failover Threshold	1
* AAA Timeout Threshold	-1
* Preferred Host	slc02xfd
Logout URL	
Logout Callback URL	/oam_logout_success
Logout Redirect URL	http://appdev-x64-rhel5-p15.us.or
Logout Target URL	

User Defined Parameters	proxySSLHeaderVar=IS_SSL
	URLInUTF8Format=true
	client_request_retry_attempts=1
	inactiveReconfigPeriod=10

* Sleep for	60
Cache Pragma Header	no-cache
Cache Control Header	no-cache
Debug	☐
IP Validation	☐
Deny On Not Protected	✓
Allow Management Operations	☐

   For the logout URL, you can add an optional `end_url` parameter to redirect the browser to a final location after users sign out. To redirect back to Studio, configure `end_url` to point to the OHS host and port.

   For example:

```
http://oamserver.us.mycompany.com:14100/oam/server/logout?end_url=http:/
/studiohost.us.company.com:7777/
```

9. To save the configuration, click **Save**.

Completing and testing the SSO integration

The final step in setting up the SSO integration is to add the OHS server host name and port to `portal-ext.properties`.

To complete and test the SSO configuration:

1. In `portal-ext.properties`, add the following lines:

```
web.server.host=ohsHostName
web.server.http.port=ohsPortNumber
```

   Where:

   - *`ohsHostName`* is the fully qualified domain name (FQDN) of the server where OHS is installed. The name must be resolvable by Studio users.

     For example, you would use `webserver01.company.com`, and not `webserver01`.

     You need to specify this even if OHS is on the same server as Studio.

- `ohsPortNumber` is the port number used by OHS.

2. Restart Studio.

   Make sure to completely restart the browser to remove any cookies or sessions associated with the Studio user login you used earlier.

3. Navigate to the Studio URL. The Oracle Access Manager SSO form is displayed.
4. Enter your SSO authentication credentials.

   You are logged in to Studio.

   As you navigate around Studio, make sure that the browser URL continues to point to the OHS server and port.

Part III

Customizing Studio

Chapter 9

Changing the Look and Feel of Studio

Users with CSS expertise can customize the look and feel of the Studio application.

About customizing the Studio look and feel

Studio allows you to customize the Studio CSS and use your own images.

Note that updating the CSS is not recommended, and you should only attempt to do this type of customization if you are very familiar with cascading style sheets.

When replacing images, if you only want to replace the image and don't want to have to update the CSS, then you should make sure that the images are the same size.

If you have a clustered instance of Studio, also be sure to make the same changes on all of the Studio instances.

Location of the Studio CSS and images

The CSS and images that control the Studio look and feel are located in the `html\css\eid-default` directory.

For Tomcat deployments, the directory is in `endeca-portal\webapps\ROOT`.

For WebLogic deployments, the directory is embedded in `endeca-portal-<versionNumber>.war`, which in turn is embedded in `endeca-portal-weblogic-<versionNumber>.ear`.

The directory contains:

- `endeca-skin.css` – The main CSS file for Studio. The CSS selectors are split among multiple CSS files. This file contains pointers to those files.
- `endeca-skin-`*`idNumber`*`.css` - The CSS files containing the CSS selectors. *`idNumber`* is a generated identifier.
- `images\extjs-default-images` – Contains third-party images used on the Studio UI.
- `images\liferay-default-images` – Contains third-party images used on the Studio UI.
- `images\oracle-default-images` – Contains our custom images to display the UI. The images are grouped by function.

Updating the Studio CSS and images for a Tomcat instance

For a Tomcat instance, you can go directly to the `css\eid-default` directory.

To update the CSS and images for a Tomcat instance:

1. Stop Studio.
2. Navigate to `endeca-portal\webapps\ROOT`.
3. Update the CSS and images.
4. Restart Studio.

   If needed, clear the browser cache in order to see the changes on the Studio UI.

Updating the Studio CSS and images for a WebLogic instance

For a WebLogic instance, before you can update the files, you need to extract the `css/eid-default` directory from the .ear file.

To extract and update files in WebLogic:

1. Stop Studio.
2. Open `endeca-portal-weblogic-<versionNumber>.ear`.
3. In the .ear file, navigate to `endeca-portal-<versionNumber>.war`.
4. Open `endeca-portal-<versionNumber>.war`.
5. In the .war file, navigate to `html/css/eid-default`.
6. Update the CSS and images as needed.
7. Resave `endeca-portal-<versionNumber>.war`, then resave this .war file in `endeca-portal-weblogic-<versionNumber>.ear`.
8. Redeploy `endeca-portal-weblogic-<versionNumber>.ear` in Weblogic.

   The method used to deploy the .ear file will vary based on whether you are in a development or a production environment.

   Once the file finishes deploying, the CSS and images are updated.

   If needed, clear the browser cache in order to see the changes on the Studio UI.

Chapter 10

Security Extensions to Studio

You may require more than the default data source role-based security discussed in the *Studio User's Guide*. If so, you can customize the automated filtering of data from the Oracle Endeca Server (based on user profile details such as the user's role or group association) by creating a custom Security Manager.

Security Manager class summary

A Security Manager is a concrete class that implements `com.endeca.portal.data.security.MDEXSecurityManager`.

Class Summary Item	Item value or Description
Abstract base class	`com.endeca.portal.data.security.MDEXSecurityManager`
Default implementation class	`com.endeca.portal.data.DefaultMDEXSecurityManager`
Description	Handles pre-execution query modification based on the user, role, or group-based security configuration of filters.

Class Summary Item	Item value or Description
Default implementation behavior	The default Security Manager implementation uses the following properties: • `securityEnabled`. If the value is not present, then `securityEnabled` defaults to `false`. • `securityFilters`. DataSourceFilters are the only supported type of `securityFilter`. • `rolePermissions` • `inheritSecurity`. If the data source has a parent, then `inheritSecurity` defaults to `true`. Otherwise, the value defaults to `false`. • `parentDataSource` These properties are defined in data source configurations in order to apply role-based security filters to queries issued to the Endeca Server backing a given data source. Users are assigned to Studio roles in the **Control Panel**. The related associations are made available to every component throughout the user's session. For each data source, the Security Manager maintains an internal map of security filters to always apply to queries issued during that user's session.

Creating a new Security Manager

The Studio Component SDK includes Windows and Linux batch scripts for creating a new Security Manager.

To create a new Security Manager project:

1. In a terminal, change your directory to `endeca-extensions` within the Component SDK's root directory (normally called `components`).
2. Run one of the following commands:
   - On Windows: `.\create-mdexsecuritymanager.bat <your-security-manager-name>`
   - On Linux: `./create-mdexsecuritymanager.sh <your-security-manager-name>`

   This command creates a `your-security-manager-name` directory under `endeca-extensions`. This directory is an Eclipse project that you can import directly into Eclipse, if you use Eclipse as your IDE.

   This directory also contains a sample implementation that you can use to help understand how the Security Manager can be used. The sample implementation is essentially identical to the default implementation of the Security Manager used by Studio.

Implementing a new Security Manager

Your Security Manager must implement the `applySecurity` method.

There are two versions of the `applySecurity` method, one of which your Security Manager must implement:

```
public void applySecurity(PortletRequest request, MDEXState mdexState, Query query) throws
MDEXSecurityException;
```

The Query class in this signature is `com.endeca.portal.data.Query`. This class provides a simple wrapper around an `ENEQuery`.

Using the Security Manager

In order to use your Security Manager, you must specify a new class for Studio to pick up and use in place of the default Security Manager implementation.

The `your-security-manager-name` directory you created contains an ant build file. The `ant deploy` task places a .jar file containing your Security Manager into the `portal/tomcat-<`*`version`*`>/lib/ext` directory.

To configure Studio to use your new class:

1. From the Studio menu, section **Control Panel**.
2. In the **Information Discovery** section of the **Control Panel** navigation panel, select **Framework Settings**.
3. Change the value of the `df.mdexSecurityManager` property to the full name of your class, similar to following example:
   ```
   df.mdexSecurityManager = com.endeca.portal.extensions.YourSecurityManagerClass
   ```
4. Click **Update Settings**.
5. Restart Studio so the change can take effect. You may also need to clear any cached user sessions.

Chapter 11

Managing Data Source State in Studio

Studio allows you to define your own interaction model for data sources by creating a custom State Manager. For information on the default interaction model between related data sources, see the *Studio User's Guide*.

About the State Manager interface

The State Manager controls how data sources interact during updates and query construction.

State Manager Interface Item	Item Value or Description
Interface (required)	`com.endeca.portal.data.MDEXStateManager`
Abstract base class (optional)	`com.endeca.portal.data.AbstractMDEXStateManager`
Default implementation class	`com.endeca.portal.data.DefaultMDEXStateManager`
Description	Handles: • Updating a data source with a new query state (called from `DataSource.setQueryState(QueryState newState)`) • Retrieving the current query state from a data source (called from `DataSource.getQueryState()`) • Resetting a data source's query state to its initial state (called from `DataSource.resetQueryState()`) • Retrieving a copy of the data source's initial state without resetting the data source (called from `DataSource.getInitialQueryState()`)

State Manager Interface Item	Item Value or Description
Default implementation behavior	The default State Manager implementation uses the `ParentDataSource` property from the data source configuration to propagate state changes throughout the hierarchy of data source relationships. When a component changes the query state of its data source, that modification is applied to: • The parent data source • All of the children of the parent data source This is recursive, applying all the way up and back down an ancestor tree. Configuring a hierarchy of data source relationships allows application developers to create more advanced interfaces, such as a tabbed result set where a single **Guided Navigation** component controls the query state for **Results Table** components on different tabs.

Creating a new State Manager

The `endeca-extensions` directory of the Component SDK includes scripts for creating a State Manager project on either Windows or Linux.

To create a new State Manager project:

1. In a terminal, change to the `endeca-extensions` directory within the Component SDK's root directory (normally called `components`).
2. Run one of the following commands:
   - On Windows: `.\create-mdexstatemanager.bat` <*your-state-manager-name*>
   - On Linux: `./create-mdexstatemanager.sh` <*your-state-manager-name*>

   This command creates a <*your-state-manager-name*> directory under `endeca-extensions`. This directory is an Eclipse project. If you use Eclipse as your IDE, you can import the project directly into Eclipse.

   The directory also contains a sample implementation, which is essentially identical to the default implementation of the State Manager used by Studio. You can use this sample implementation to help understand how to use the State Manager.

Implementing a State Manager

Custom State Managers implement the `MDEXStateManager` interface. There are methods for updating, retrieving, and resetting the data source query state.

Recommendations for implementing

To create a custom State Manager, you must at minimum implement the `com.endeca.portal.data.MDEXStateManager` interface. The recommended approach is to extend `com.endeca.portal.data.AbstractMDEXStateManager`, which in turn implements `MDEXStateManager`.

You also should extend `com.endeca.portal.data.AbstractMDEXStateManager`, which in turn implements `MDEXStateManager`. The `AbstractMDEXStateManager` abstract class contains the useful utility method `addEventTrigger(PortletRequest, MDEXState)`.

The default state manager implementation is `com.endeca.portal.data.DefaultMDEXStateManager`. The Studio Component SDK creates state managers that extend `DefaultMDEXStateManager`, because they will work without any modification. If you want your custom state manager to inherit some of the default functionality, you can extend `DefaultMDEXStateManager` instead of `AbstractMDEXStateManager`.

Required methods

Your State Manager must implement the following methods:

```
public void handleStateUpdate(PortletRequest request, MDEXState mdexState, QueryState newQueryState)
throws QueryStateException;

public QueryState handleStateMerge(PortletRequest request, MDEXState mdexState) throws
QueryStateException;

public void handleStateReset(PortletRequest request, MDEXState mdexState) throws QueryStateException;

public QueryState handleStateInitial(PortletRequest request, MDEXState mdexState) throws
QueryStateException;
```

Method Name	Method Description
`handleStateUpdate()`	Called when a component calls `DataSource.setQueryState(qs)`. This method should eventually call `mdexState.setQueryState()`. However, it is not required to make this call if it determines that the `MDEXState's QueryState` should not change. If the data source state is changed by `handleStateUpdate()`, you must mark the affected data sources. To mark the data sources, you call the `addEventTrigger(PortletRequest request, MDEXState ds)` method, passing in the request object and any `MDEXState` objects that are changed.

Method Name	Method Description
`handleStateMerge()`	Called when a component calls `DataSource.getQueryState()`. You are expected to return the `QueryState` that the component should get access to for the data source represented by the `mdexState`, taking into account any data source relationships or other aspects of your `State Manager` that might affect the query state.
`handleStateReset()`	Called when a component calls `DataSource.resetQueryState()`. This method returns the data source to the "initial state" defined by your state manager. The default implementation (`DefaultMDEXStateManager`) clears all query functions from the data source except those defined in the `baseFunctions` key of the data source's .json file, and similarly updates all parent and child data sources. If the data source state changes while it is being reset, you must mark the affected data sources. To mark the data sources, you call the `addEventTrigger(PortletRequest request, MDEXState ds)` method, passing in the request object and any `MDEXState` objects that are changed.
`handleStateInitial()`	Called when a component calls `DataSource.getInitialQueryState()`. This method returns a copy of the data source's initial state as defined by your state manager. The default implementation (`DefaultMDEXStateManager`) returns a `QueryState` with query functions made up of the union of the `baseFunctions` from: • The current data source • All of the current data source's parents

Using the State Manager

In order to use your State Manager, you must specify a new class for Studio to pick up and use in place of the default State Manager implementation.

The `<your-state-manager-name>` directory you created contains an ant build file. The `ant deploy` task places a `.jar` file containing your State Manager into the `portal/tomcat-<version>/lib/ext` directory.

To configure Studio to use your State Manager:

1. From the Studio menu, select **Control Panel**.
2. In the **Information Discovery** section of the **Control Panel** navigation panel, select **Framework Settings**.

3. Change the value of `df.mdexStateManager` property to the full name of your class, similar to following example:

   ```
   df.mdexStateManager = com.endeca.portal.extensions.YourStateManagerClass
   ```

4. Click **Update Settings**.
5. Restart Studio so the change can take effect. You may also need to clear any cached user sessions.

Chapter 12

Installing and Using the Component SDK

The Studio Component SDK is a packaged development environment that you can use to add or modify components and layout templates.

Note: To develop components using the Studio Component SDK, you must work with the Tomcat bundle version of Studio. Once the components are completed, you can deploy them on any platform.

Software and licensing requirements for component development

To develop custom components, you need the following software and licenses.

Software requirements

In addition to the Studio Component SDK, component development requires the following software:

- Eclipse
- JDK 1.5 or above
- Apache Ant 1.7.1 or higher

Ext JS license requirement

Studio uses *Ext JS* in its components and in the default components created by its SDK.

The Oracle Endeca Information Discovery license does not bundle licensing for Ext JS.

Therefore, customers developing components with Ext JS must either purchase their own development licenses from Ext JS, or remove Ext JS and develop components without using that Javascript framework.

About obtaining junit.jar for component unit tests

If you are planning to create unit tests for your custom components, you will need to first obtain `junit.jar`.

The Component SDK can use JUnit for unit tests, but does not come with the `junit.jar` file.

Downloading and configuring the Component SDK

The Studio Component SDK .zip file is available from the Studio Media Pack.

To download and install the Component SDK:

1. From the Oracle Endeca Information Discovery Studio Media Pack for Windows or Linux, download the Oracle Endeca Information Discovery Studio SDK.

   For Windows, the file is `EIDStudio<`*`version`*`>_WindSDK.zip`.

   For Linux, the file is `EIDStudio<`*`version`*`>_LinuxSDK.zip`.

2. Unzip the file into a separate directory.

   Note: Do not install the Component SDK in a directory path that contains spaces.

3. Within the Component SDK:

   (a) Create the following file:

   `components/build.<`*`user`*`>.properties`

   In the file name, `<`*`user`*`>` is the user name that you use used to log in to the current machine.

   (b) Within that file, add the following property:

   ```
   portal.base.dir=<absolute_path_to_portal>
   ```

   The value `<`*`absolute_path_to_portal`*`>` is the path to the `endeca-portal` directory for the Studio instance.

   Note: On Windows, backslashes in paths must be escaped. For example, use:

   ```
   portal.base.dir=C:\\my_folder\\endeca-portal
   ```

   instead of:

   ```
   portal.base.dir=C:\my_folder\endeca-portal
   ```

   (c) In the `shared/` directory, create a `shared.properties` file .

   (d) In `shared.properties`, add the following property:

   ```
   portal.base.dir=<absolute_path_to_portal>
   ```

   The value `<`*`absolute_path_to_portal`*`>` is the path to the `endeca-portal` directory for the Studio instance.

   Note: On Windows, you must escape backslashes in paths. For example, use:

   ```
   portal.base.dir=C:\\my_folder\\endeca-portal
   ```

   instead of:

   ```
   portal.base.dir=C:\my_folder\endeca-portal
   ```

Configuring Eclipse for component development

Before using the Component SDK to develop Studio components in Eclipse, you need to create two Eclipse classpath variables.

Note: Depending on your version of Eclipse, the steps below may vary slightly.

To configure the Eclipse classpath variables for Studio component development:

1. In Eclipse, go to **Window>Preferences>Java>Build Path>Classpath Variables**.
2. Create two new variables:

Name	Path
`DF_GLOBAL_LIB`	Path to the application server global library. Example: `C:/endeca-portal/tomcat-<`*`version`*`>/lib`
`DF_PORTAL_LIB`	Path to the Studio ROOT Web application library. Example: `C:/endeca-portal/tomcat-<`*`version`*`>/webapps/ROOT/WEB-INF/lib`

Once these variables have been created, the components generated by the Component SDK can be imported into Eclipse.

Developing a new component

Here is a high-level overview of the component development process.

To develop a new Studio component:

1. Create the component.
2. Import the project in Eclipse.
3. Build and test the new component.

Creating a new component

New Studio components are extensions of the `EndecaPortlet` class.

To create a new component:

1. At a command prompt, navigate to the Component SDK directory, and from there to `components/portlets`.
2. Run the command:

```
create.bat <component-name-no-spaces> "<ComponentDisplayName>"
```

For example:

```
create.bat johns-test "John's Test Component"
```

In the command, the first argument is the component name. The component name:

- Cannot have spaces.
- Cannot include the string `-ext`, because it causes confusion with the ext plugin extension. For example, `my-component-extension` would not be a valid name.
- Has the `-portlet` automatically appended to the name. For example, if you set the name to `johns-test`, the name will actually be `johns-test-portlet`.

The second argument is intended to be a more human-friendly display name. The display name can have spaces, but if it does, it must be enclosed in quotation marks.

Importing the project in Eclipse

Before beginning component development, you have to import the component project you just created into Eclipse.

To import the Studio Component SDK project you just created into Eclipse:

1. Within Eclipse, choose **File>Import>General>Existing Projects into Workspace**.
2. As the root directory from which to import, select the directory where you installed the Component SDK.

   You should see multiple projects to import.
3. Import the components you need to work with.

   If your components depend on shared library projects located within the `/shared` directory, import those as well.

   Note: It takes some time for projects to build after they are imported.

Building and testing your new component

Next, you can build your new component in Eclipse and ensure that it is available in Studio.

To build your new component in Eclipse:

1. In your new project, open the `build.xml` file at the top level.
2. In the outline view, right-click the deploy task and select **Run as...>Ant Build**.

   Note: This step is only necessary if you do not have **Build Automatically** checked in the Eclipse **Project** menu.
3. If Studio is not already running, start Studio and log in.
4. Look at the Studio logs to confirm that the component was picked up successfully.

5. To test your new component within Studio:
   (a) From the Studio menu, select **Add Component**.
   (b) In the **Add Component** dialog, expand the **Sample** category.

   Your component should be listed in that category.
   (c) To add the new component to the Studio page, drag and drop it from the **Add Component** dialog.

Adding and removing components from the WebLogic .ear file

If you have installed Studio on Oracle WebLogic Server, then you can also add the component to the deployed .ear file, so that it will be deployed automatically the next time you deploy the file, for example when installing a production instance after you have completed testing on a development instance.

To add components to and remove components from the WebLogic .ear file:

1. To add a custom component to the .ear file:
   (a) Copy your component to the `<LIFERAY_HOME>/deploy` directory.
   (b) After the component has been processed and moved to the `<`*`LIFERAY_HOME`*`>/weblogic-deploy` directory, undeploy the .ear file.
   (c) Add the processed component .war file to the root of the zipped .ear file.
   (d) In the .ear file, add an entry for the new component to `META-INF/application.xml`.
2. To remove a component from the .ear file:
   (a) Remove the component .war file from the root of the .ear file.
   (b) In the .ear file, remove the component entry from `META-INF/application.xml`.

Modifying the Component SDK build enhancements

The `build.xml` file in the root directory of each component created by the Component SDK contains properties that control whether to include the build enhancements.

By default, these properties are:

```
<property name="shared.libs" value="endeca-common-resources,endeca-discovery-taglib" />
<property name="endeca-common-resources.includes" value="**/*" />
<property name="endeca-common-resources.excludes" value="" />
```

The properties control the following behavior:

Property	Description
`shared.libs`	Controls which projects in the `shared/` directory to include in your component. These shared projects are compiled and included as `.jar` files where appropriate.

Property	Description
`endeca-common-resources.includes`	Controls which files in the `shared/endeca-common-resources` project are copied into your component. The default value is `"**/*"`, indicating that all of the files are included, These files provide: • AJAX enhancements (`preRender.jspf` and `postRender.jspf`) • The ability to select a different data source for the component (`dataSourceSelector.jspf`)
`endeca-common-resources.excludes`	Controls which files from the `shared/endeca-common-resources` project are excluded from your component. By default, the value is `""`, indicating that no files are excluded. If your component needs to override any of these files, you must use this build property to exclude them. If you do not exclude them, your code will be overwritten.

The `includes` and `excludes` properties can be specified for any shared library, for example:

```
<property name="endeca-discovery-taglib.includes" value="**/*" />
<property name="endeca-discovery-taglib.excludes" value="" />
```

Chapter 13

Working with QueryFunction Classes

Studio provides a set of `QueryFunction` classes to allow you to filter and query data. You can also create and implement your own `QueryFunction` classes.

Provided QueryFunction filter classes

Studio provides the following filter classes. Filters are used to change the current query state. They can be used in the definition of a Studio data source, or called by a custom component.

The available filter classes are:

- `DataSourceFilter`
- `RefinementFilter`
- `NegativeRefinementFilter`
- `RangeFilter`
- `SearchFilter`

Note that the examples below use the syntax for calling the filters from a component. For details on configuring filters in a data source definition, see the *Studio User's Guide*.

DataSourceFilter

Uses an EQL snippet to provide the filtering.

When used in a data source definition, a `DataSourceFilter` is a permanent filter designed to be used for security purposes.

The available properties are:

Property	Description
`filterString`	The EQL snippet containing the filter information. For a `DataSourceFilter`, this would be the content of a `WHERE` clause for an EQL statement. For details on the EQL syntax, see the *Oracle Endeca Server EQL Guide*.

For example, to filter data to only show records from the Napa Valley region with a price lower than 40 dollars:

```
ExpressionBase expression = dataSource.parseLQLExpression("Region='Napa Valley' and P_Price<40");
DataSourceFilter dataSourceFilter = new DataSourceFilter(expression);
```

RefinementFilter

Used to filter data to include only those records that have the provided attribute values. End users can remove `RefinementFilter` refinements.

The properties for a `RefinementFilter` are:

Property	Description
`attributeValue`	String The attribute value to use for the refinement. For a managed attribute, this is the value ID.
`attributeKey`	String The attribute key. Identifies the attribute to use for the refinement.
`multiSelect`	`AND |OR | NONE` For multi-select attributes, how to do the refinement if the filters include multiple values for the same attribute. If set to `AND`, then matching records must contain all of the provided values. If set to `OR`, then matching records must contain at least one of the provided values. If set to `NONE`, then multi-select is not supported. Only the first value is used for the refinement.

In the following example, the data is refined to only include records that have a value of 1999 for the Year attribute.

```
RefinementFilter refinementFilter = new RefinementFilter("1999", "Year");
```

NegativeRefinementFilter

Used to filter data to exclude records that have the provided attribute value. End users can remove `NegativeRefinementFilter` refinements.

The properties for a `NegativeRefinementFilter` are:

Property	Description
`attributeValue`	String The attribute value to use for the refinement.
`attributeKey`	String The attribute key. Identifies the attribute to use for the refinement.
`attributeType`	`BOOLEAN | STRING | DOUBLE | LONG | GEOCODE | DATETIME | TIME | DURATION` The type of value to use for the refinement. The default is `STRING`. If the attribute is a standard attribute of a type other than string, then you must provide the type.
`attributeValueName`	String Optional. The value to display on the **Breadcrumbs** component for the refinement. If you do not provide a value for `attributeValueName`, then the **Breadcrumbs** component displays the value of `attributeValue`. You may want to provide a separate display value if the selected attribute is a managed attribute for which the value names are different from the actual stored value.
`ancestors`	String List Optional. The display names of the ancestor values to display on the **Breadcrumbs** component. You would most likely want to provide ancestor values when selecting a managed attribute value from a value hierarchy.

In the following example, the data is refined to only include records that do NOT have a value of Washington for the Region attribute. Because Region is a string attribute, no other configuration is needed.

```
NegativeRefinementFilter negativeRefinementFilter
= new NegativeRefinementFilter("Region", "Washington");
```

In the following example, the data is refined to only include records that do NOT have a value of 1997 for the P_Year attribute. Because P_Year is not a string attribute, the attribute type LONG is specified.

```
NegativeRefinementFilter negativeRefinementFilter
= new NegativeRefinementFilter("P_Year", "1997", "LONG");
```

In the following example, the data is refined to only include records that do NOT have Caterer as the value for the Outlet attribute. The values for Outlet are stored as codes, so a display name to use for the breadcrumb is provided. Also, Outlet is a hierarchical attribute, and the breadcrumb indicates that Caterer is a subcategory of Nonstore Retailers under the category Retail Sales.

```
List<String> ancestors = new ArrayList<String>();
ancestors.add("Retail Sales");
ancestors.add("Nonstore Retailers");
NegativeRefinementFilter negativeRefinementFilter
= new NegativeRefinementFilter("Outlet", "454210", "Caterer", ancestors);
```

RangeFilter

Used to filter data to include only those records that have attribute values within the specified range. End users can remove `RangeFilter` refinements.

The properties for a `RangeFilter` are:

Property	**Description**
`attributeKey`	String The attribute key. Identifies the attribute to use for the filter.
`rangeOperator`	`LT` | `LTEQ` |`GT` |`GTEQ`| `BTWN` |`GCLT` |`GCGT` | `GCBTWN` The type of comparison to use. • `LT` - Less than • `LTEQ` - Less than or equal to • `GT` - Greater than • `GTEQ` - Greater than or equal to • `BTWN` - Between. Inclusive of the specified range values. • `GCLT` - Geocode less than • `GCGT` - Geocode greater than • `GCBTWN` - Geocode between
`rangeType`	`NUMERIC` | `CURRENCY` | `DATE` | `GEOCODE` The type of value that is being compared.
`value1`	Numeric The value to use for the comparison. For `BTWN`, this is the low value for the range. For the geocode range operators, the origin point for the comparison.

Property	Description
`value2`	Numeric For a `BTWN`, this is the high value for the range. For `GCLT` and `GCGT`, this is the value to use for the comparison. For `GCBTWN`, this is the low value for the range.
`value3`	Numeric Only used for the `GCBTWN` operator. The high value for the range.

In the following example, the data is refined to only include records where the value of P_Score is a number between 80 and 100:

```
RangeFilter rangeFilter
= new RangeFilter("P_Score", RangeType.NUMERIC, RangeOperator.BTWN, "80", "100");
```

SearchFilter

Used to filter the data to include records that have the provided search terms. End users can remove `SearchFilter` refinements.

The properties for a `SearchFilter` are:

Property	Description
`searchInterface`	String Either the name of the search interface to use, or the name of an attribute that is enabled for text search.
`terms`	String The search terms.
`matchMode`	`ALL` | `PARTIAL` | `ANY` | `ALLANY` | `ALLPARTIAL` | `PARTIALMAX` | `BOOLEAN` The match mode to use for the search.
`enableSnippeting`	boolean Whether to enable snippeting. Optional. If not provided, the default is `false`.

Property	Description
`snippetLength`	int The number of characters to include in the snippet. Required if `enableSnippeting` is `true`. To enable snippeting, set `enableSnippeting` to `true`, and provide a value for `snippetLength`.

In the following example, the filter uses the "default" search interface to search for the terms "California" and "red". The matching records must include all of the search terms. Snippeting is supported, with a 100-character snippet being displayed.

```
SearchFilter.Builder builder = new SearchFilter.Builder("default", "California red");
builder.matchMode(MatchMode.ALL);
builder.enableSnippeting(true);
builder.snippetLength(100);
SearchFilter searchFilter = builder.build();
```

Provided QueryConfig functions

Studio provides the following `QueryConfig` functions, used to manage the results returned by a query. These are more advanced functions for component development.

Each `QueryConfig` function generally has a corresponding function in `DiscoveryServiceUtils` to get the results.

`QueryConfig` functions are specific to a component. Because of this, `QueryConfig` functions should never be persisted to a data source using `setQueryState()`, as this would affect all of the components bound to that data source. Instead, `QueryConfig` functions should only be added to a component's local copy of the `QueryState` object.

The available `QueryConfig` functions are:

- `AttributeValueSearchConfig`
- `BreadcrumbsConfig`
- `ExposeRefinement`
- `LQLQueryConfig`
- `NavConfig`
- `RecordDetailsConfig`
- `ResultsConfig`
- `ResultsSummaryConfig`
- `SearchAdjustmentsConfig`
- `SearchKeysConfig`
- `SortConfig`

AttributeValueSearchConfig

Used for typeahead in search boxes. For example, used in Guided Navigation to narrow down the list of available values for an attribute.

`AttributeValueSearchConfig` has the following properties:

Property	Description
`searchTerm`	String The term to search for in the attribute values.
`maxValuesToReturn`	int (optional) The maximum number of matching values to return. If you do not provide a value, then the default is 10.
`attribute`	String (optional) The attribute key for the attribute in which to search. Use the `attribute` property to search against a single attribute. To search against multiple attributes, use `searchWithin`.
`searchWithin`	List<String> (optional) A list of attributes in which to search for matching values.
`matchMode`	`ALL|PARTIAL|ANY|ALLANY|ALLPARTIAL|PARTIALMAX|BOOLEAN` (optional) The match mode to use for the search.
`relevanceRankingStrategy`	String (optional) The name of the relevance ranking strategy to use during the search.

The following example searches for the term "red" in the WineType attribute values:

```
AttributeValueSearchConfig attributeValueSearchConfig
= new AttributeValueSearchConfig("red", "WineType");
```

BreadcrumbsConfig

Used to return the breadcrumbs associated with the query. Allows you to specify whether to display the full path for hierarchical attribute values.

`BreadcrumbsConfig` has the following property:

Property	Description
`returnFullPath`	boolean (optional) For a hierarchical managed attribute, whether to return the full path to the selected value. The default is `true`, indicating to return the full path. To not return the full path, set this to `false`.

This example returns the breadcrumbs, but does not return the full path for hierarchical managed attributes:

```
BreadcrumbsConfig breadcrumbsConfig = new BreadcrumbsConfig(false);
```

ExposeRefinement

Affects results from a `NavConfig` function. Used to implement Guided Navigation. Controls whether to display available attributes within groups, and whether to display available refinements for attributes.

`ExposeRefinement` has the following properties:

Property	Description
`dimValId`	String The ID of the selected attribute value. You would provide an attribute value ID if you were displaying the next level of available values in a managed attribute hierarchy.
`dimensionId`	String The name of the attribute. You must provide at least one `dimValId` or `dimensionId`.
`ownerId`	String (optional) The ID of the associated `NavConfig` instance. If not provided, then uses the first `NavConfig` instance.
`dimExposed`	boolean (optional) Whether to display the available values for the attribute, to the number specified in `maxRefinements`. The default is `true`.

Property	Description
`exposeAll`	boolean (optional) Whether to display the complete list of available values. For example, on the **Guided Navigation** component, would indicate whether the "More..." link is selected. The default is `false`.
`maxRefinements`	int (optional) The maximum number of available values to display. The default is `1000`.
`groupKey`	String (required) The name of a group.
`groupExposed`	boolean (optional) Whether to display all of the attributes in the specified group. The default is `true`.

The following example shows the available attributes for the Flavors attribute within the Characteristics group.

```
ExposeRefinement exposeRefinement = new ExposeRefinement("/", "Flavors", "Characteristics");
```

LQLQueryConfig

Executes an EQL query on top of the current filter state.

`LQLQuery` has the following property:

Property	Description
`LQLQuery`	AST The EQL query to add. To retrieve the AST from the query string, call `DataSource.parseLQLQuery`.

The following example retrieves the average of the P_Price attribute grouped by Region:

```
Query query
= dataSource.parseLQLQuery("return mystatement as select avg(P_Price) as avgPrice group by Region",
true);
LQLQueryConfig lqlQueryConfig = new LQLQueryConfig(query);
```

NavConfig

Used to retrieve a navigation menu, such as in the Guided Navigation component.

`NavConfig` has the following properties:

Property	Description
`exposeAllRefinements`	boolean Whether to display all of the available values for the attributes. Determines the initial state of the menu. The associated `ExposeRefinement` function is then applied. The default is false.
`List<RefinementGroupConfigs>`	List of groups for which to return the available attributes. If no `RefinementGroupConfigs` are specified, no attribute groups or attributes are returned.

The following examples returns attributes in the Source and Characteristics groups:

```
List<RefinementGroupConfig> refinementGroups = new ArrayList<RefinementGroupConfig>();
RefinementGroupConfig source = new RefinementGroupConfig();
source.setName("Source");
source.setExpose(true);
refinementGroups.add(source);
RefinementGroupConfig characteristics = new RefinementGroupConfig();
characteristics.setName("Characteristics");
characteristics.setExpose(true);
refinementGroups.add(characteristics);
NavConfig navConfig = new NavConfig();
navConfig.setRefinementGroupConfig(refinementGroups);
```

RecordDetailsConfig

Sends an attribute key-value pair to assemble the details for a selected record. The complete set of attribute-value pairs must uniquely identify the record.

`RecordDetailsConfig` has the following property:

Property	Description
`recordSpecs`	List<RecordSpec> Each new `RecordDetailsConfig` is appended to the previous `RecordDetailsConfig`.

The following example sends the value of the P_WineID attribute:

```
List<RecordSpec> recordSpecs = new ArrayList<RecordSpec>();
recordSpecs.add(new RecordSpec("P_WineID", "37509"));
RecordDetailsConfig recordDetailsConfig = new RecordDetailsConfig(recordSpecs);
```

ResultsConfig

Used to manage the returned records. Allows for paging of the records.

`ResultsConfig` has the following properties:

Property	Description
`recordsPerPage`	long The number of records to return at a time.
`offset`	long (optional) The position in the list at which to start. The very first record is at position 0. For example, if `recordsPerPage` is 10, then to get the second page of results, the offset would be 10.
`columns`	String[] (optional) The columns to include in the results. If not specified, then the results include all of the columns.
`numBulkRecords`	int (optional) The number of records to return. Overrides the value of `recordsPerPage`.

The following example returns a selected set of columns for the third page of records, where each page contains 50 records:

```
ResultsConfig resultsConfig = new ResultsConfig();
resultsConfig.setOffset(100);
resultsConfig.setRecordsPerPage(50);
String[] columns = {"Wine_ID", "Name", "Description", "WineType", "Winery", "Vintage"};
resultsConfig.setColumns(columns);
```

ResultsSummaryConfig

Gets the number of records returned from a query.

```
ResultsSummaryConfig resultsSummaryConfig = new ResultsSummaryConfig();
```

SearchAdjustmentsConfig

Returns "Did you mean" and auto-correction items for a search.

```
SearchAdjustmentsConfig searchAdjustmentsConfig = new SearchAdjustmentsConfig();
```

SearchKeysConfig

Returns the list of available search interfaces.

```
SearchKeysConfig searchKeysConfig = new SearchKeysConfig();
```

SortConfig

Used to sort the results of a query. Used in conjunction with `ResultsConfig`.

`SortConfig` has the following properties:

Property	Description
`ownerId`	String (optional) The ID of the `ResultsConfig` that this `SortConfig` applies to. If not provided, uses the default `ResultsConfig` ID. If you configure a different ID, then you must provide a value for `ownerId`.
`property`	String The attribute to use for the sort.
`ascending`	boolean Whether to sort in ascending order. If set to `false`, then the results are sorted in descending order.

For example, with the following `SortConfig`, the results are sorted by the P_Score attribute in descending order:

```
SortConfig sortConfig = new SortConfig("P_Score", false);
```

Creating a custom QueryFunction class

The Component SDK directory includes scripts for creating new `QueryFunction` classes.

Note: Before you can create `QueryFunction` classes, you must install the Component SDK, which is a separate download. See *Downloading and configuring the Component SDK on page 58*.

To create a new `QueryFilter` or `QueryConfig` class:

1. In a terminal window, change to the `endeca-extensions` subdirectory of the Component SDK's root directory (normally called `components`).
2. Run the appropriate command to create the `QueryFilter` or `QueryConfig` class.

   To create a `QueryFilter` class:

Operating System	Command Syntax
Windows:	`.\create-queryfilter.bat` *`<your-query-filter-name>`*
Linux:	`./create-queryfilter.sh` *`<your-query-filter-name>`*

To create a `QueryConfig` class:

Operating System	Command Syntax
Windows:	`.\create-queryconfig.bat` <*your-query-config-name*>
Linux:	`./create-queryconfig.sh` <*your-query-config-name*>

The command creates in the `endeca-extensions` directory a new directory for the `QueryFilter` or `QueryConfig` class:

- For a `QueryFilter`, the directory is <*your-query-filter-name*>`-filter`.
- For a `QueryConfig`, the directory is <*your-query-config-name*>`-config`.

This directory is an Eclipse project that you can import directly into Eclipse, if you use Eclipse as your IDE.

It contains an empty sample implementation of a `QueryFilter` or `QueryConfig`. This has no effect on `QueryState` in its original form.

The skeleton implementation creates source files that:

- Extend either `QueryFilter` or `QueryConfig`.
- Create stubs for the `applyToDiscoveryServiceQuery`, `toString`, and `beforeQueryStateAdd` methods.

  `applyToDiscoveryServiceQuery` and `toString` are required methods that you must implement.

  `beforeQueryStateAdd` is an optional method to verify the query state before the function is added. This method is used to prevent invalid query states such as duplicate refinements.
- Create a no-argument, protected, empty constructor. The protected access modifier is optional, but recommended.
- Create a private member variable for logging.

Implementing a custom QueryFunction class

After you create your new `QueryFunction` class, you then implement it.

To implement your new `QueryFunction`, you must:

- Add private filter or configuration properties.
- Create getters and setters for any filter properties you add.
- Define a no-argument constructor (protected access modifier optional, but recommended).
- Optionally, implement the `beforeQueryStateAdd(QueryState state)` method to check the current query state before the function is added.

Deploying a custom QueryFunction class

Before you can use your new `QueryFunction`, you must deploy it to Studio.

The directory that you created for the new `QueryFilter` or `QueryConfig` contains an ant build file.

The `ant deploy` task places a `.jar` file containing the custom `QueryFunction` into the `endeca-portal/tomcat-<version>/lib/ext` directory.

Note: If you are not using the default portal bundle, put the new `QueryFunction.jar` into the container's global classpath.

To deploy the new `QueryFunction`:

1. Run the ant build.
2. Restart Studio.

   The portal picks up the new class.

After you deploy your custom `QueryFunction`, you can use it in any component.

Adding the custom QueryFunction .jar file to your Eclipse build path

If you are using Eclipse as your IDE, you need to add the new `.jar` file to the build path of your custom component.

To add the new `.jar` file to your Eclipse build path:

1. Right-click the project, then select **Build Path>Configure Build Path**.
2. Click the **Libraries** tab.
3. Click **Add Variable**.
4. Select **DF_GLOBAL_LIB**.

   You should have added this variable when you set up the SDK.
5. Click **Extend**.
6. Open the `ext/` directory.
7. Select the `.jar` file containing your custom `QueryFunction`.
8. Click **OK**.

After adding the `.jar` file to the build path, you can import the class, and use your custom `QueryFilter` or `QueryConfig` to modify your `QueryState`.

Obtaining query results

The `Results` class is used to represent results of queries.

You must add the relevant `QueryConfigs` to a component in order to specify the types of results it needs.

```
QueryState query = getDataSource(request).getQueryState();
query.addFunction(new NavConfig());
QueryResults results = getDataSource(request).execute(query);
```

You can then get the underlying API results and do whatever manipulation is required by your component.

```
Results discoveryResults = results.getDiscoveryServiceResults();
```

Before executing, you can also make other local modifications to your query state by adding filters or configurations to your query:

```
QueryState query = getDataSource(request).getQueryState();
query.addFunction(new ResultsConfig());
ExpressionBase expression =
getDataSource(request).parseLQLExpression("Region = 'Midwest'");
query.addFunction(new SelectionFilter(expression));
QueryResults results = getDataSource(request).execute(query);
```

When you need to update a data source's state to update all of the associated components, you must use `QueryState` instances.

```
DataSource ds = getDataSource(request);
QueryState query = ds.getQueryState();
ExpressionBase expression =
getDataSource(request).parseLQLExpression("Region = 'Midwest'");
query.addOperation(new SelectionFilter(expression));
ds.setQueryState(query);
```

Index