Endeca® Discovery Framework

Installation Guide

Contents

Preface....7
About this guide....7
Who should use this guide....7
Conventions used in this guide....7
Contacting Endeca Customer Support....8

Chapter 1: Before you install....9
Overview of the Endeca Discovery Framework....9
Interaction with Liferay Portal....9
Features available in this version....10
System requirements....11
Compatibility with other Endeca products....12
MDEX 7 Early Access edition support....12
Upgrading from a previous version of the Discovery Framework....12
Obtaining more information....13

Chapter 2: Installing the Discovery Framework....15
Installing the full Discovery Framework package....15
Downloading the Endeca Discovery Framework software for the full installation....15
Installing the Windows Tomcat bundle....16
Installing the Linux Tomcat bundle....18
Installing the Discovery Framework on Tomcat 5.5....19
Installing the Discovery Framework on the WebSphere Application Server version 6.1....23
Installing the Discovery Framework on the WebSphere Application Server version 7....31
Discovery Framework interaction with IAP 6.1.4....39
Installing the components-only package....39
Downloading the Endeca Discovery Framework software for a component installation....40
Prerequisites to installing a Discovery Framework component release....40
Installing the component-only package on any supported version of Tomcat....41
Installing the components-only package on WebSphere....41
Downloading the Endeca Discovery Framework documentation....42

Chapter 3: About data source configuration....45
About data sources....45
Data source syntax....45
About the sample MDEX Engine data sources....47
Specifying a default data source....47
Adding data sources to the Discovery Framework....48
Changing an Endeca component's data source....48
Configuring aggregated records in a data source....48
Configuring the Discovery Framework to connect to a secured MDEX Engine....49
Data source role-based security....50
About data source relationships....50
Provided QueryFunction classes in the Discovery Framework....51
Implementing new QueryFunction classes....55
Creating a QueryFunction class....55
Implementing the QueryFunction class....56
Using your custom QueryFunction....56
Adding the new .jar file to your Eclipse build path....57
Obtaining data source results....57

Chapter 4: About Discovery Framework logging....59
Modifying logging in your Discovery Framework components....59
Adjusting the logging verbosity level in the Control Panel....59
Modifying portal-log4j-ext.xml....61
Setting up logging for your Discovery Framework application....61

About log4j.properties files....62
Learning about log4j....62

Chapter 5: Getting started with the Discovery Framework....63
Starting the Discovery Framework....63
Accessing the Control Panel....63
About Framework Settings....64
Adding Endeca standard components 66

Chapter 6: Other installation tasks....67
Using a different database....67
Overview of switching to a different database....67
Installing Corda....68
Obtaining the Corda software....68
About the Corda Server servlet....68
About the Discovery Framework implementation of Corda....69
Deploying the Corda Server servlet in an application server....69
Updating the Chart component with changes to the Corda Server servlet....71
Troubleshooting Corda....73
Uninstalling the Discovery Framework....74

Copyright and disclaimer

Endeca Discovery Framework Installation Guide • December 2010

Version 1.4

Preface

Endeca® Latitude applications guide people to better decisions by combining the ease of search with the analytic power of business intelligence. Users get self-service access to the data they need without needing to specify in advance the queries or views they need. At the same time, the user experience is data driven, continuously revealing the salient relationships in the underlying data for them to explore.

The heart of Endeca's technology is the MDEX Engine.™ The MDEX Engine is a hybrid between an analytical database and a search engine that makes possible a new kind of Agile BI. It provides guided exploration, search, and analysis on any kind of information: structured or unstructured, inside the firm or from external sources.

Endeca Latitude includes data integration and content enrichment tools to load both structured and unstructured data. It also includes the Discovery Framework, a set of tools to configure user experience features including search, analytics, and visualizations. This enables IT to partner with the business to gather requirements and rapidly iterate a solution.

About this guide

This guide contains installation instructions for setting up the Endeca Discovery Framework on Windows and Linux.

The Discovery Framework enables rapid configuration of dashboard applications that offer the highly interactive Guided Navigation® user experience across a full range of structured and unstructured enterprise data.

The Discovery Framework is easy to deploy and ideal for the agile development of enterprise-quality applications. Due to component-based nature of the Discovery Framework, these applications are simple to control, adapt, and extend. It provides granular layout and configuration control to enable users to manage and personalize their own experiences.

The Discovery Framework consists of an enterprise-class portal framework and a library of UI components that embody best practices in Endeca applications. In addition, it includes a Component SDK, which is a packaged development environment for portlets, themes, layout templates, and other portal element. Endeca has modified Liferay's version of its Plugins SDK to include the Endeca enhancements, such as the `EndecaPortlet` core class.

Who should use this guide

This guide is intended for developers who are building applications using the Endeca Discovery Framework on Windows or Linux.

Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: ¬

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

Contacting Endeca Customer Support

The Endeca Support Center provides registered users with important information regarding Endeca software, implementation questions, product and solution help, training and professional services consultation as well as overall news and updates from Endeca.

You can contact Endeca Standard Customer Support through the Support section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*.

Chapter 1

Before you install

This section provides an overview of the Endeca Discovery Framework, system requirements, and other information you need to know before installing.

Overview of the Endeca Discovery Framework

The Discovery Framework enables rapid configuration of dashboard applications that offer the highly interactive Guided Navigation® user experience across a full range of structured and unstructured enterprise data.

The Discovery Framework is easy to deploy and ideal for the agile development of enterprise-quality applications. Due to component-based nature of the Discovery Framework, these applications are simple to control, adapt, and extend. It provides granular layout and configuration control to enable users to manage and personalize their own experiences. The Discovery Framework consists of an enterprise-class portal framework and a library of UI components that embody best practices in Endeca applications.

About the Component SDK

The Component SDK is a packaged development environment for portlets, themes, layout templates, and other portal element. Endeca has modified Liferay's version of its Plugins SDK to include the Endeca enhancements, such as the `EndecaPortlet` core class. The installation and use of the Component SDK is covered in the *Discovery Framework Extension Guide*.

Interaction with Liferay Portal

The Discovery Framework is built upon the Liferay Portal Enterprise Edition.

Liferay Portal is an open-source JSR-286 portal technology. The Discovery Framework extends basic Liferay functionality to provide enhanced user management, security, and cross-component interaction, as well as performance-optimized communication with Endeca MDEX Engines.

This version of the Discovery Framework is built upon Liferay Portal 5.2 Enterprise Edition Service Pack 4.

Features available in this version

The Discovery Framework 1.4 provides a core set of application capabilities designed to address the most common requirements for search applications.

The Discovery Framework contains the following Endeca components:

- The **Attribute Settings** component allows you to modify the display names of attributes in your data sources. Note that this component appears in the Liferay Control Panel.
- The **Bookmarks** component allows you to save the navigation state and component state of a given component so that you can return to them.
- The **Breadcrumbs** component provides breadcrumb navigation aid functionality.
- The **Chart** component lets you access Corda-based Analytics charting.
- The **Cross Tab** component provides a table that allows end users to perform comparisons and identify trends across several cross sections of data.
- The **Data Sources** component allows you to view configured data sources and test the connection to them. In addition, you can reload updated configuration based on edits you have made on disk. Note that this component appears in the Liferay Control Panel.
- The **Data Source Bindings** component allows you to associate different configured data sources with selected components in a single operation, rather than on a per-component basis. Note that this component appears in the Liferay Control Panel.
- The **Find Similar** component allows you to locate and view a set of records that share common attributes with a particular record of interest.
- The **Framework Settings** component provides access to state, security, and other settings. Note that this component appears in the Liferay Control Panel.
- The **Guided Navigation** component provides Endeca Guided Navigation functionality.
- The **Metrics Bar** component allows you to view a set of numerical or text values that summarize various aspects of the underlying data. In some situations, this component can be used to indicate when a metric has crossed a specified threshold value.
- The **Performance Metrics** component displays information about component and MDEX Engine query performance. Note that this component appears in the Liferay Control Panel.
- The **Range Filter** component allows you to add and modify range filters.
- The **Record Details** component displays all of the properties for the record in question.
- The **Results Table** component provides a simple interface for displaying results, along with an example of view transitions.

Note: Version 1.4 also includes a Beta version of an enhanced **Results Table** component.

- The **Sample** component provides developers with a template from which they can build their own custom components.
- The **Searchbox** component provides searchbox functionality.
- The **Tabbed Component Container** allows you to create a tabbed interface within a region of a page and then store different components on various tabs.
- The **Advanced Visualization** component provides Xcelsius dashboarding within your application.

Note: For more information about components, see the *Discovery Framework Component Catalog*.

Liferay components in the Discovery Framework

The Discovery Framework includes a set of Liferay Portal content management components that allow you to do things like publish HTML content or embed an external Web site or application in a component.

For an overview of Liferay Web content management, see *here*. For more details, see this Liferay *blog post*, or search the *Liferay Portal* site.

The Discovery Framework includes Liferay's **Languages** component, which lets you change the server locale.

System requirements

The Endeca Discovery Framework version 1.4 has the following requirements:

Hardware requirements

The hardware requirements for the Discovery Framework 1.4 are the same as those for Endeca MDEX Engine version 6.1.x. For details, see the *Endeca MDEX Engine Installation Guide*.

Supported operating systems

The Discovery Framework 1.4 is supported on the same Windows and Linux operating systems as the Endeca MDEX Engine version 6.1.x, with the exception noted below. For details, see the *Endeca MDEX Engine Installation Guide*.

Important: The Discovery Framework is not supported on Sparc Solaris.

Software requirements

The Discovery Framework is a Web-based application that runs in an application server.

- Supported browsers: Firefox 3.6, Internet Explorer 8 (with compatibility mode disabled)

  Tip: Firefox is recommended.

  Important: Running Internet Explorer 8 in compatibility mode is not supported.

  Note: The Discovery Framework supports Adobe Flash 10.0.

- Supported application servers: Tomcat 6, Tomcat 5.5, WebSphere Application Server (WAS) 6.1.0.1 or higher, WebSphere Application Server (WAS) 7
- Supported Java versions: Tomcat 6 is supported with Sun Java 6; Tomcat 5.5 is supported with Sun Java 5; WAS 6.1 with IBM Java 5, WAS 7 with IBM Java 6

  Important: For Sun Java 6, update 18 or greater is required.

- Supported database systems: MySQL 5.1, DB2 9.5

Alternative database support

The Liferay Portal server uses a relational database to store configuration and state. By default, Liferay uses Hypersonic, but this is not recommended for production use due to performance issues. Endeca tests the Discovery Framework on MySQL and DB2. However, many other databases are expected to work. Customers should feel free to use any database, including shared systems they may already have in place. As with application servers, customers who choose to deploy on un-tested databases

will always be supported on any issue that can be traced back to core Discovery Framework code and can be reproduced on a supported database.

The *Discovery Framework Installation Guide* combined with the *Liferay Portal Administrator's Guide* provides detailed instructions on how to switch to another database system.

Compatibility with other Endeca products

This document assumes that you already have a running MDEX Engine at which you can point the Discovery Framework.

The Endeca Discovery Framework version 1.4 is compatible with the following Endeca components:

- MDEX Engine 6.1.x (up to 6.1.4) or 7.0.1 Early Access (see below)

Note: For best stability and performance, if you are using Discovery Framework with the MDEX Engine 6.1.3, ensure that you obtain the latest patches for that release.

- Platform Services 6.0.1 or 6.1
- Developer Studio 6.0.1 or 6.1
- Deployment Template 3.1 or 3.2

Important: The Discovery Framework is supported on Dgraph deployments only. It is not supported for use with the Agraph.

MDEX 7 Early Access edition support

This release of the Discovery Framework includes limited capabilities for use with MDEX Engine Release 7.0.1. These capabilities are available only to users who have access to MDEX 7.0.1 Early Access edition and can connect the Discovery Framework to an MDEX 7.0.1 data source.

If you do not have access to MDEX 7.0.1 Early Access edition, you can safely ignore any mention of it in the Discovery Framework 1.4 documentation.

MDEX 7.0.1 features are in an Early Access state. The interfaces and behavior of these Early Access features may change in later releases of MDEX 7 or of the Discovery Framework, based on information gathered during the Early Access period. These capabilities are not supported for use in production.

Connecting to an MDEX 7.0.1 data source

For information about connecting to an MDEX 7.0.1 data source, see the topic "Connection to an MDEX 7.0.1 data source," which is located in the section "About data sources."

Upgrading from a previous version of the Discovery Framework

For information about upgrading from a previous version of the Discovery Framework, see the *Discovery Framework Migration Guide*.

Note: You can only have one version of the Discovery Framework installed on your machine at a time.

Obtaining more information

Because the Discovery Framework is built upon the Liferay Portal, you can access Liferay's documentation for more information about how to perform administrative tasks.

Specifically, the *Liferay Portal Administrator's Guide* provides extensive information about installing, configuring, and maintaining a portal. To access a free PDF download of this guide, go to *http://www.liferay.com* and navigate to Documentation.

Liferay developer resources

In addition to its formal administrator documentation, Liferay offers developer assistance in the form of blogs, wikis, and forums. To access this, go to *http://www.liferay.com* and navigate to Community.

The Endeca Developer Network (EDeN)

You can obtain more information about the Discovery Framework and other Endeca products at the Endeca Developer Network (EDeN) at *http://eden.endeca.com*. In particular, EDeN's Discovery Framework forum provides discussions for technical and business users of Endeca's Discovery Framework and its components, including topics such as development, extension, deployment, and configuration.

Additional Endeca documentation

The Discovery Framework `doc` directory contains the following documents:

- The *Discovery Framework Migration Guide*, which provides information on upgrading from earlier versions of the Discovery Framework.
- The *Discovery Framework Component Catalog*, which provides an overview of each of the standard components.
- The *Discovery Framework Extension Guide* and Discovery Framework javadoc, both aimed at developers using the Discovery Framework.

Chapter 2

Installing the Discovery Framework

This section contains the Discovery Framework installation procedures for the supported application servers. It covers both full portal installations and component-only installations.

Installing the full Discovery Framework package

After downloading the Discovery Framework software, you can install it on your development server.

There are five options for installing this release of the Discovery Framework:

- Discovery Framework with the Windows Tomcat bundle. This is based on Tomcat 6 and Java 1.6.
- Discovery Framework with the Linux Tomcat bundle. This is based on Tomcat 6 and Java 1.6.
- Discovery Framework as a standalone application on Tomcat 5.5 application server.
- Discovery Framework as a standalone application on Websphere Application Server 6.1.
- Discovery Framework as a standalone application on Websphere Application Server 7.

Note: The following steps will deploy the portal using the default embedded Hypersonic database, which is not intended for production use. In production, you must deploy using an alternate database. More information about this process can be found in chapter 6 of this guide. Briefly, deploying an alternate database can be accomplished by modifying the `portal-ext.properties` file to specify the appropriate JDBC connection information for the desired database. Alternatively, you can follow the instructions in the *Liferay Portal Administrator's Guide* to set up a JDBC provider and data source in your application server, and configure the `portal-ext.properties` to look up the data source by JNDI name.

Important: The Discovery Framework requires the Endeca Theme in order to start up and allow you to enter a license key. Even if you do not intend to use the Endeca Theme in production, you should not uninstall the Endeca Theme (`endeca-theme-<version>.war`) from the `endeca-portal\deploy` directory.

Downloading the Endeca Discovery Framework software for the full installation

You can download the Endeca Discovery Framework from the Downloads section of the Endeca Developer Network (EDeN).

To download the Discovery Framework software:

1. If you have not previously done so, establish a Support account with download access through the Support section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*. This enables the Endeca Support and Customer Care groups to track which versions of the software you are using.
2. Navigate through the EDeN site as follows:
   a) On the EDeN homepage, click **Downloads**.
   b) On the **Tools and Utilities** page, find the **Product Downloads** section and click **View and download purchased products**.
   c) On the **Product Downloads** page, find and click **Discovery Framework**.
   d) In the **Current Releases** table, click **Discovery Framework <*version*>**.

   The **Product Download** page contains links to all available Discovery Framework packages.
3. Download the appropriate Discovery Framework zip files, depending on your installation environment:
   - To install the Tomcat 6 bundle for Windows, download `endeca-portal-<version>.zip` and `components-<version>.zip` to your development server.
   - To install the Tomcat 6 bundle for Linux, download `endeca-portal-<version>.tgz` and `components-<version>.zip` to your development server.
   - To install Discovery Framework for Tomcat 5.5 or the Websphere Application Server version 6.1 or 7, download `endeca-portal-<version>.war.zip`, `endeca-portal-dependencies-<version>.zip`, and `components-<version>.zip`.

Important: If you are doing a component-only installation, see the separate download list in the section "Installing the components-only package."

Note: For instructions on downloading and installing the Corda package (which is only required if you plan to use the **Chart** component), see the section "Installing Corda."

Installing the Windows Tomcat bundle

This topic provides the steps for installing the Discovery Framework Windows Tomcat bundle on your development server. In this version Tomcat 6 and the JVM 1.6 are embedded.

Note: Among the data sources in your Discovery Framework application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Specifying a default data source," located in the chapter "About data source configuration."

To install the Discovery Framework Tomcat bundle:

1. Unzip `endeca-portal-<version>.zip` to the directory of your choice.
2. Extract the `.war` files from `components-<version>.zip` and place them into the `endeca-portal\deploy` directory. The `.war` files go in the root of `endeca-portal\deploy`. There should be no subdirectories.

Note: This directory already contains themes, hooks, and layouts required by the portal. It is safe to overwrite these files with the versions in `components-<version>.zip`.

3. If the environment variables `CATALINA_HOME` or `JAVA_HOME` are already set, update them to point to your newly installed Tomcat directory and a valid 1.6 JRE. For example, `set CATALI¬ NA_HOME=C:\path\to\endeca-portal\tomcat-6.0.18`. (If you do not have these environment variables set, you can leave them un-set.)
4. Start the portal's Tomcat instance by running `endeca-portal\tomcat-6.0.18\bin\start¬ up.bat`.

   Note: Server startup can take several minutes. You can follow the log messages to ascertain when the process is complete. Do not shut down the Tomcat window while the Discovery Framework is running.

5. Go to the portal (`http://localhost:8080/`) in your browser, and log in using the following default credentials:

Option	Description
Email address	test@endeca.com
Password	test

6. Upon first use, enter the license key, as described in the topic that appears later in this section.
7. Optionally, you can set up *log4j* logging. `log4j` provides configurable, Java-based logging in an open-source utility.

   Note: For more information about Discovery Framework logging, see Chapter 4.

Changing the context root for the Windows Tomcat bundle

Optionally, you can change the context root after installing the Windows Tomcat bundle.

To change the context root:

1. Rename `endeca-portal\tomcat-6.0.18\conf\Catalina\localhost\ROOT.xml` file to `<context root>.xml`. For example, if your context root is `sales`, the file name should be `sales.xml`.

   For multi-level context paths, separate the name with #. For example, for a context path of `/sales/east`, the file name should be `sales#east.xml`.
2. Modify the XML file created in the previous step as needed:
   - For a root context: `<Context path="" />`
   - For a context of `/sales`: `<Context path="/sales"/>`
   - For a context of `/sales/east`: `<Context path="/sales/east"/>`
3. Rename the `endeca-portal\tomcat-6.0.18\webapps\ROOT` directory to `endeca-portal\tomcat-6.0.18\webapps\<context root>`. For multi-level context paths, use a multi-level path like the following: `endeca-portal\tomcat-6.0.18\webapps\sales\east`.
4. Edit the `endeca-portal\portal-ext.properties` file. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you used above. However, do not include a trailing slash in the `portal.ctx` value. For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

Do not use this value:

```
portal.ctx=/mycompany/portal/
```

Installing the Linux Tomcat bundle

This topic provides the steps for installing the Discovery Framework Linux Tomcat bundle on your development server. In this version Tomcat 6 and the JVM 1.6 are embedded.

Note: Among the data sources in your Discovery Framework application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Specifying a default data source," located in the chapter "About data source configuration."

To install the Discovery Framework Tomcat bundle:

1. Extract `endeca-portal-<version>.tgz` to the directory of your choice.
2. Extract the `.war` files from `components-<version>.zip` and place them into the `endeca-portal/deploy` directory. The `.war` files go in the root of `endeca-portal/deploy`. There should be no subdirectories.

   Note: This directory already contains themes, hooks, and layouts required by the portal. It is safe to overwrite these files with the versions in `components-<version>.zip`.

3. If the environment variables `CATALINA_HOME` or `JAVA_HOME` are already set, update them to point to your newly installed Tomcat directory and a valid 1.6 JRE. (If the `JAVA_HOME` environment variable is not set, you must set it.)
4. Start the portal's Tomcat instance by running `endeca-portal/tomcat-6.0.18/bin/start-up.sh`.

   Note: Server startup can take several minutes. You can follow the log messages to ascertain when the process is complete. Do not shut down the Tomcat window while the Discovery Framework is running.

5. Upon first use, enter the license key, as described in the next topic.
6. Go to the portal (`http://localhost:8080/`) in your browser, and log in using the following default credentials:

Option	Description
Email address	test@endeca.com
Password	test

7. Optionally, you can set up *log4j* logging. `log4j` provides configurable, Java-based logging in an open-source utility.

   Note: For more information about Discovery Framework logging, see Chapter 3.

Changing the context root in the Linux Tomcat bundle

Optionally, you can change the context root used by your Discovery Framework application.

To change the context root:

1. Rename `endeca-portal/tomcat-6.0.18/conf/Catalina/localhost/ROOT.xml` file to `<context root>.xml`. For example, if your context root is `sales`, the file name should be `sales.xml`.

   For multi-level context paths, separate the name with #. For example, for a context path of `/sales/east`, the file name should be `sales#east.xml`.

2. Modify the XML file created in the previous step as needed:

   - For a root context: `<Context path="" />`
   - For a context of `/sales`: `<Context path="/sales"/>`
   - For a context of `/sales/east`: `<Context path="/sales/east"/>`

3. Rename the `endeca-portal/tomcat-6.0.18/webapps/ROOT` directory to `endeca-portal/tomcat-6.0.18/webapps/<context root>`. For multi-level context paths, use the multi-level path here: `endeca-portal/tomcat-6.0.18/webapps/ROOT` directory to `endeca-portal/tomcat-6.0.18/webapps/sales/east`.
4. Edit the `endeca-portal/portal-ext.properties` file. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you used above. However, do not include a trailing slash in the `portal.ctx` value. For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

Entering the license key

The Discovery Framework is build upon Liferay Portal Enterprise Edition. When the Discovery Framework is initially deployed and started, the first user to access the application is prompted for a license key.

To license your version of the Discovery Framework:

1. Download the license key from the Discovery Framework section of EDeN.
2. In the **Liferay Portal license** dialog box, enter the key in the **License Key** field.
3. Click **Save**. The portlet updates the dialog box with more detailed information about the license key.
4. Click **Save** again. The dialog box closes and you can proceed with your work.

Installing the Discovery Framework on Tomcat 5.5

You can deploy the Discovery Framework as a standalone application on Tomcat 5.5.

These instructions assume that you have obtained the `apache-tomcat-5.5.x.zip` or `tar.gz` file from the *Apache Foundation* but that you have not yet installed it. The rest of these instructions will refer to the installation directory as `apache-tomcat-5.5.x`, leaving off the minor version number. Before following the steps here, consult the *Liferay Portal Administrator's Guide*, which contains portal deployment instructions and examples for Tomcat 5.5.

Note: The examples in this section are based on a Windows server Tomcat deployment. If you are installing on Linux, the steps will be similar, though you will need to substitute Linux binaries and paths. Where there is a significant difference, this is called out.

Note: Among the data sources in your Discovery Framework application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Specifying a default data source," located in the chapter "About data source configuration."

High-level overview of Tomcat 5.55 deployment

This topic provides an overview of the steps you need to take to deploy the Discovery Framework as a standalone application on Tomcat 5.5.

Details on each of these steps appear in the topics that follow.

To deploy the Discovery Framework on Tomcat 5.5:

1. Install Tomcat and deploy the Discovery Framework dependency libraries.
2. Modify Tomcat configuration to work with the Discovery Framework.
3. Deploy and start the Discovery Framework application.

Installing Tomcat 5.5 and deploying the dependency libraries

The Discovery Framework requires the deployment of several Java libraries.

To install the Tomcat software and deploy the Discovery Framework dependency libraries:

1. Create an `endeca-portal` directory. This will be the home directory for your Discovery Framework installation.
2. Create an `apache-tomcat-<version>` directory under the `endeca-portal` directory.
3. Unzip `apache-tomcat-5.5.x.zip` into `endeca-portal/apache-tomcat-5.5.x`, where `x` indicates the minor version number.

   Unzipping this file creates much of the directory structure mentioned below.
4. Unzip `endeca-portal-dependencies-<version>.zip` into a temporary directory.

   This zip file contains a collection of `.jar` files and other dependency files.
5. From the temporary directory, copy the following `.jar` files into the `endeca-portal/apache-tomcat-5.5.x/common/endorsed` directory:

   ```
   log4j.jar
   log4j.properties.jar
   ccpp.jar
   jutf7.jar
   ```
6. Under the `endeca-portal/apache-tomcat-5.5.x/common/lib` directory, create an `ext` directory.
7. From the temporary directory you created in step 4, copy the following `.jar` files into the `endeca-portal/apache-tomcat-5.5.x/common/lib/ext` directory that you just created:

   ```
   activation.jar
   annotations.jar
   commons-lang.jar
   cs_bindings.jar
   ```

```
cxf-2.2.8.jar
cxf-rt-databinding-jaxb-2.2.8.jar
endeca-images.jar
endeca-portal.jar
endeca_navigation.jar
ext-service.jar
geronimo-activation_1.1_spec-1.0.2.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-jaxws_2.1_spec-1.0.jar
geronimo-saaj_1.3_spec-1.0.1.jar
geronimo-stax-api_1.0_spec-1.0.1.jar
geronimo-ws-metadata_2.0_spec-1.1.2.jar
hsql.jar
jabsorb.jar
jackson-core-lgpl-1.6.2.jar
jackson-mapper-lgpl-1.6.1.jar
jaxb-api-2.1.jar
jaxb-impl-2.1.7.jar
jms.jar
jsr173_1.0_api.jar
jta.jar
jtds.jar
mail.jar
mysql.jar
portal-kernel.jar
portal-service.jar
portlet.jar
postgresql.jar
stax-1.2.0.jar
wsdl4j-1.6.2.jar
wstx.jar
XmlSchema-1.4.3.jar
```

Modifying Tomcat configuration to work with the Discovery Framework

Before proceeding further, you must modify some Tomcat configuration files.

1. In the `endeca-portal/apache-tomcat-5.5.x/bin/` directory, modify `catalina.bat` (on Windows) or `catalina.sh` (on Linux) by adding the JAVA_OPTS line. This line should appear under the line `Execute The Requested Command` as follows:

   ```
   set JAVA_OPTS=%JAVA_OPTS% -Xmx1024m -XX:MaxPermSize=256m -Dfile.encod¬
   ing=UTF8 -Duser.timezone=GMT -Dorg.apache.catalina.loader.WebappClassLoad¬
   er.ENABLE_CLEAR_REFERENCES=false
   ```

   This increases the memory size for the server and establishes security configuration for the Discovery Framework.

2. Modify the `endeca-portal/apache-tomcat-5.5.x/conf/catalina.properties` file as follows to add the `ext` directory to the common class loader:

   ```
   common.loader=
        ${catalina.home}/common/classes,\
        ...\
        ${catalina.home}/common/lib/ext/*.jar
   ```

3. To deploy the Discovery Framework in the root context, create a new file called `ROOT.xml` and place it in `endeca-portal/apache-tomcat-5.5.x/conf/Catalina/localhost/`. To deploy the Discovery Framework into any other context, create a new file called `<context root>.xml` and place it in `endeca-portal/apache-tomcat-5.5.x/conf/Catalina/localhost/`.

For multi-level context paths, separate the name with #. For example, for a context path of `/sales/east`, the file name should be `sales#east.xml`.

4. Modify the XML file created in the previous step as needed:
   - For a root context: `<Context path="" />`
   - For a context of `/sales`: `<Context path="/sales"/>`
   - For a context of `/sales/east`: `<Context path="/sales/east"/>`
5. Rename the `endeca-portal\apache-tomcat-5.5.x\webapps\ROOT` directory to `endeca-portal\apache-tomcat-5.5.x\webapps\<context root>`. For multi-level context paths, use the multi-level path here: `endeca-portal\apache-tomcat-5.5.x\webapps\ROOT` directory to `endeca-portal\apache-tomcat-5.5.x\webapps\mycompany\sales`.
6. To support UTF-8 URI encoding, edit the `server.xml` file located in the `endeca-portal/apache-tomcat-5.5.x/conf` directory as follows:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
   <Connector
      port="8080"
      maxHttpHeaderSize="8192"
      maxThreads="150"
      minSpareThreads="25"
      maxSpareThreads="75"
      enableLookups="false"
      redirectPort="8443"
      acceptCount="100"
      connectionTimeout="20000"
      disableUploadTimeout="true"
      URIEncoding="UTF-8"
   />
```

Deploying and starting the Discovery Framework application

Once Tomcat configuration is complete, the Discovery Framework application can be deployed and started.

To deploy and start the Discovery Framework application:

1. Delete the contents of the `endeca-portal/apache-tomcat-5.5.x/webapps/ROOT` directory.

   This directory contains the standard Web application that is installed with Tomcat by default. We will replace this standard web application with the Discovery Framework application in the next step.
2. Unzip `endeca-portal-<version>.war.zip` into a temporary directory.

   This zip file contains the `endeca-portal-<version>.war` file and the `copyright.txt` file.
3. Read the `copyright.txt` file and then save it to the location of your choice.
4. Unzip the contents of `endeca-portal-<version>.war` into the `endeca-portal/apache-tomcat-5.5.x/webapps/ROOT` directory.
5. Copy the `portal-ext.properties` file from the temporary directory you created for the `endeca-portal-dependencies-<version>.zip` file in step 4 to the `endeca-portal` directory.
6. Edit the `endeca-portal\portal-ext.properties` file. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you set earlier. However, do not include a trailing slash in the `portal.ctx` value. For example, use this value:

```
portal.ctx=/mycompany/portal
```

Do not use this value:

```
portal.ctx=/mycompany/portal/
```

7. Under the `endeca-portal` directory, create a `data` directory, and then create an `endeca-data-sources` directory below that.
8. Create a data source to place in the `endeca-portal/data/endeca-data-sources` directory. For information about data sources, see the *Discovery Framework Extension Guide*. In addition, you can reference the sample data source files, which are located in the `endeca-data-sources` directory in the temporary directory you created for the `endeca-portal-dependencies-<version>.zip` file in a previous step.
9. Start the portal's Tomcat instance by running `endeca-portal\tomcat-5.5.x\bin\start-up.bat`.

Note: Server startup can take several minutes. You can follow the log messages to ascertain when the process is complete. Do not shut down the Tomcat window while the Discovery Framework is running.

10. Go to the portal (`http://localhost:8080/`) in your browser, and log in using the following default credentials:

Option	Description
Email address	test@endeca.com
Password	test

11. Upon first use, enter the license key, as described in the topic that appears later in this section.

Entering the license key

The Discovery Framework is build upon Liferay Portal Enterprise Edition. When the Discovery Framework is initially deployed and started, the first user to access the application is prompted for a license key.

To license your version of the Discovery Framework:

1. Download the license key from the Discovery Framework section of EDeN.
2. In the **Liferay Portal license** dialog box, enter the key in the **License Key** field.
3. Click **Save**. The portlet updates the dialog box with more detailed information about the license key.
4. Click **Save** again. The dialog box closes and you can proceed with your work.

Installing the Discovery Framework on the WebSphere Application Server version 6.1

You can deploy the Discovery Framework as a standalone application on WebSphere Application Server (WAS) version 6.1. (6.1.0.1 or higher).

Before following the steps here, consult the *Liferay Portal Administrator's Guide*, which contains portal deployment instructions and examples for WebSphere Application Server 6.1.

Note: The examples in this section are based on a Linux server WAS deployment. If you are installing on Windows, the steps will be similar, though you will need to substitute Windows executables and paths. In certain examples, backslashes are used to escape the dollar sign ($)

character on Linux, because the shell would otherwise attempt a variable substitution for this character. These backslashes should not be required on a Windows system.

Note: Among the data sources in your Discovery Framework application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Specifying a default data source," located in the chapter "About data source configuration."

High-level overview of Websphere Application Server 6.1 deployment

This topic provides an overview of the steps you need to take to deploy the Discovery Framework on WAS 6.1.

Details on each of these steps appear in the topics that follow.

To deploy the Discovery Framework on WAS 6.1:

1. Deploy dependency `.jar` files. The exact list of required files and destination directories appears below.
2. Start (or restart) the WAS server.
3. Install the Discovery Framework `.war` file as an enterprise application.
4. Edit and deploy `portal-ext.properties`.
5. Create the `endeca-data-sources/*.json` data source configuration files.

   For more information, see the section "About data sources."
6. Install the Endeca theme, portlet components, and other framework `.war` files.
7. Start the Discovery Framework enterprise application, entering the license key upon first use.
8. Optionally, repeat step 6 for any additional plugins you want to add.

Deploying Discovery Framework dependency libraries on WAS 6.1

The Discovery Framework requires the deployment of several Java libraries.

These libraries are deployed to a global class loader, making them available to multiple applications.

To deploy the Discovery Framework dependency libraries on WAS 6.1:

1. Unzip the `.jar` files found in `endeca-portal-dependencies-<version>.zip`.
2. Upload the following list of `.jar` files from the `.zip` file to the WAS server's external library directory. (For example, if WAS is installed in `/usr/local/WAS/AppServer`, you would deploy the selected `.jar` files into `/usr/local/WAS/AppServer/lib/ext/`.)

```
annotations.jar
ccpp.jar
commons-lang.jar
cs_bindings.jar
cxf-2.2.8.jar
cxf-rt-databinding-jaxb-2.2.8.jar
endeca-images.jar
endeca-portal.jar
endeca_navigation.jar
ext-service.jar
geronimo-activation_1.1_spec-1.0.2.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-jaxws_2.1_spec-1.0.jar
```

```
geronimo-saaj_1.3_spec-1.0.1.jar
geronimo-stax-api_1.0_spec-1.0.1.jar
geronimo-ws-metadata_2.0_spec-1.1.2.jar
hsql.jar
jabsorb.jar
jackson-core-lgpl-1.6.2.jar
jackson-mapper-lgpl-1.6.1.jar
jaxb-api-2.1.jar
jaxb-impl-2.1.7.jar
jsr173_1.0_api.jar
log4j.jar
portal-kernel.jar
portal-service.jar
portlet.jar
slf4j-api.jar
slf4j-log4j12.jar
stax-1.2.0.jar
wsdl4j-1.6.2.jar
wstx.jar
XmlSchema-1.4.3.jar
```

3. Do one of the following, depending on the version of the MDEX Engine you are using with your Discovery Framework application:
   - If you are using MDEX 6 data sources, move the file `portlet.jar` from the WAS server's external library directory (`/usr/local/WAS/AppServer/lib/ext/` in the example above) to `/usr/local/WAS/AppServer/java/jre/lib/ext/.`
   - If you are using MDEX 7 data sources, move the following list of `.jar` files from the WAS server's external library directory (`/usr/local/WAS/AppServer/lib/ext/` in the example above) to `/usr/local/WAS/AppServer/java/jre/lib/ext/`:

     ```
     cxf-2.2.8.jar
     geronimo-activation_1.1_spec-1.0.2.jar
     geronimo-annotation_1.0_spec-1.1.1.jar
     geronimo-jaxws_2.1_spec-1.0.jar
     geronimo-saaj_1.3_spec-1.0.1.jar
     geronimo-stax-api_1.0_spec-1.0.1.jar
     geronimo-ws-metadata_2.0_spec-1.1.2.jar
     jaxb-api-2.1.jar
     jaxb-impl-2.1.7.jar
     portlet.jar
     wsdl4j-1.6.2.jar
     XmlSchema-1.4.3.jar
     ```

4. Restart the WAS server so that it can pick up the newly available `.jar` files.

Extracting the standalone portal WAR on WAS 6.1

Before you can install the standalone portal WAR, you must extract it from its download package.

To extract the standalone portal WAR on WAS 6.1:

1. Unzip `endeca-portal-<version>.war.zip` into a temporary directory.
   This zip file contains the `endeca-portal-<version>.war` file and the `copyright.txt` file.
2. Read the `copyright.txt` file and then save it to the location of your choice.
3. Note the location of the `endeca-portal-<version>.war` file, as you will need to navigate to it in the next step.

Deploying the standalone portal WAR on WAS 6.1

After downloading and extracting the necessary files, you can deploy the Discovery Framework as an enterprise application in WebSphere Application Server, and then install components, themes, and other plugins as modules in that enterprise application.

The following steps document the installation procedure by using the IBM Integrated Solutions Console for a WebSphere Application Server installed and maintained without the use of the Deployment Manager, and consisting of one cell with one node and one server. The instructions may need to be adjusted for clustered environments, environments maintained with the Deployment Manager, or for environments where administration is performed by using tools like `wsadmin`, rather than the Integrated Solutions Console.

The following steps assume that no other applications are deployed in the same application server. If there are other applications, ensure that no applications are bound to context root `/` (or that any such applications are stopped during the Discovery Framework deployment). After following these steps, you will be able to adjust the context root for the Discovery Framework application, to ensure it does not conflict with other applications.

To deploy the Discovery Framework standalone portal WAR on WebSphere Application Server version 6.1:

1. Start the WAS server.
2. Log in to the WAS Integrated Solutions Console, using the appropriate administrator credentials.
3. In the WAS Integrated Solutions Console, select **Applications** > **Install New Application**.
4. Click to browse to and select the Endeca Discovery Framework WAR you downloaded earlier (`endeca-portal-<version>.war`).
5. Set the context root to the desired path for your Discovery Framework installation.

   Note: Make a note of your context root, as you will need to reference it several times during the deployment process.

6. Select **Show me all installation options and parameters**, and then click **Next**.
7. In the **Choose to generate default bindings and mappings** section, check **Generate Default Bindings**, and then select the following options:

   - **Override existing bindings**
   - **Use default virtual host name for Web and WIP modules** (enter `default_host` in the text field)

   Click **Next**.
8. In the **Select installation options** step, the default application name is **endeca-portal-*<version>*_war**. Set the application name to a more relevant name (for example, `DiscoveryFramework`). All other installation options can remain unchanged. Click **Next**.

   Note: Do not use spaces in the application name. For example, use **DiscoveryFramework** instead of **Discovery Framework**.

9. Keep the defaults for all other options and then click **Finish**.
10. Wait for installation and, if it is successful, click **Save directly to master configuration**.

Creating the Liferay Home directory for WAS 6.1

The remaining instructions in this section refer to a directory called Liferay Home. The Liferay Home directory is created relative to the user's home directory.

Manually create a Liferay Home directory (`/home/endeca/liferay/`), along with the following subdirectories:

- `/home/endeca/liferay/data`
- `/home/endeca/liferay/data/endeca-data-sources`
- `/home/endeca/liferay/websphere-deploy`

Editing the portal-ext.properties file for WAS 6.1 deployment

Before deploying your `portal-ext.properties` file, you must edit it.

Endeca's default version of `portal-ext.properties` is included in the package `endeca-portal-dependencies-<version>.zip`.

1. Open the `portal-ext.properties` file and add the following lines to the end of the file:

```
# Specify a directory where Liferay will "deploy" processed plugins.
# From this directory, WAS users will deploy WARs as modules in the
# Discovery Framework enterprise application. Replace ${liferay.home}
# with the appropriate directory path, such as /home/endeca.
#
auto.deploy.dest.dir=${liferay.home}/websphere-deploy
#
# Set this to true to enable JMX integration in
# com.liferay.portal.cache.EhcachePortalCacheManager.
#
ehcache.portal.cache.manager.jmx.enabled=false
```

Note: The destination directory (specified by the `auto.deploy.dest.dir` setting) must exist before the plugin is hot-deployed. In the above example, you must manually create the `websphere-deploy` directory if it does not exist.

2. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you used when deploying the standalone portal WAR. However, do not include a trailing slash in the `portal.ctx` value.
   For example, use this value:

```
portal.ctx=/mycompany/portal
```

   Do not use this value:

```
portal.ctx=/mycompany/portal/
```

3. Save the file.

Configuring portal-ext.properties for WAS 6.1 deployment

After you edit your `portal-ext.properties` file, there are two ways to deploy it in WAS 6.1.

- By updating the application to include the `portal-ext.properties` file.
- By uploading the `portal-ext.properties` file to the Liferay Home directory on the server.

Both methods are described in the following topics.

Updating the application to include the portal-ext.properties file on WAS 6.1

After you create the `portal-ext.properties` file, you can use the IBM Integrated Solutions Console to update the portal WAR module with the additional file.

This topic documents the use of the Integrated Solutions Console to update the Discovery Framework application, to include `portal-ext.properties` in the `endeca-portal.war` module. These steps may be performed with the `wsadmin` tool instead of the Integrated Solutions Console and may need to be adjusted for alternate WAS configurations.

Note: In order to make changes to the `portal-ext.properties` file, users will need to repeat these steps to update the application with updated versions of the `portal-ext.properties` file. In some environments, it may be more appropriate to deploy the `portal-ext.properties` file to the Liferay Home directory, where it can be updated without updating the deployed application. That option is described in the next topic.

To deploy a `portal-ext.properties` file in the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications** and select the enterprise application created when you deployed the portal WAR. Click Update.
2. Select **Replace or add a single file**.
3. Specify the path to deploy the file into the `WEB-INF/classes` directory of the portal Web application. For example: `endeca-portal-<version>.war/WEB-INF/classes/portal-ext.properties`
4. Browse to where you created the file on your computer.
5. Once the file has successfully updated, click **Save directly to master configuration**.

Uploading portal-ext.properties to Liferay Home on the server on WAS 6.1

After you create the `portal-ext.properties` file, you can manually upload it to WAS 6.1.

To manually upload the `portal-ext.properties` file:

Upload the `portal-ext.properties` file to the Liferay Home directory. For example: `/home/endeca/liferay/portal-ext.properties`.
Liferay reads these properties when the Discovery Framework application is started.

Example settings for portal-ext.properties on WAS 6.1

The Endeca installation includes a default version of `portal-ext.properties`.

This file serves as a useful starting point for configuration of the portal properties, and should be deployed to the application server according to the steps described in a previous topic.

Note: Most of the settings in the default `portal-ext.properties` file are not specific to deployment on WAS 6.1. However, the following additional settings (which you must add to the file as described in the topic "Editing the portal-ext.properties file for WAS 6.1 deployment") are important for portlet deployment on WAS:

```
auto.deploy.dest.dir=/home/endeca/liferay/websphere-deploy
ehcache.portal.cache.manager.jmx.enabled=false
```

Keep in mind that the destination directory (specified by the `auto.deploy.dest.dir` setting) must exist before the plugin is hot-deployed. In the above example, you must manually create the `websphere-deploy` directory if it does not exist.

Deploying Endeca data source configuration on WAS 6.1

To configure one or more MDEX Engines as data sources for the Discovery Framework, a JSON configuration file needs to be deployed for each MDEX Engine.

These files should be deployed relative to the Liferay Home directory. Sample data source configuration files are provided as `.json.sample` files in the `endeca-portal-dependencies-<version>.zip` file you downloaded.

To deploy Endeca data source configuration:

Upload the files to the `data/endeca-data-sources/` subdirectory.

For example: `/home/endeca/liferay/data/endeca-data-sources/default.json`

Starting the application on WAS 6.1

Once the Discovery Framework application has been deployed, and the `portal-ext.properties` file has been configured and deployed, the application needs to be started.

The following steps describe this process in the IBM Integrated Solutions Console.

To start the application:

1. Go to **Applications** > **Enterprise Application** and select the enterprise application created when you deployed the portal WAR.
2. If it is not already running, click **Start** to start it.
3. View your deployed application at the root context of the server.
4. Restart WAS 6.1.

Deploying components and other plugins in WAS 6.1

This topic describes how to deploy components, themes, hooks, and other plugins in WAS 6.1.

These plugins are located in the `components-<version>.zip` package.

About Liferay component pre-processing in WAS 7
WAS does not support the hot deployment of components. However, Liferay's deployment code must update plugins by adding necessary libraries and configuration files.

For example, Liferay's portlet deployment code adds the following important piece of configuration to a portlet component's `web.xml` file:

```
<context-param>
  <param-name>com.ibm.websphere.portletcontainer.PortletDeploymentEn¬
abled</param-name>
  <param-value>false</param-value>
</context-param>
```

This context parameter is important for WAS deployment, as it ensures that WAS's portal server does not attempt to load the new portlet, and instead allows the Discovery Framework to load the newly deployed portlet.

For this reason, Liferay must be allowed to pre-process components before they are deployed to WAS. You upload your `.war` files to Liferay's `deploy` directory so that Liferay's deployer can find and process them.

Deploying components in WAS 7
Liferay must pre-process Discovery Framework components before they can be deployed in WAS 7.

Important: The Discovery Framework requires the Endeca Theme in order to start up and allow you to enter a license key. Even if you do not intend to use the Endeca Theme in production, you should deploy the Endeca Theme (`endeca-theme-<version>.war`).

To deploy Discovery Framework components in WAS 7:

1. Copy all component `.war` files to `${liferay.home}/deploy`.
2. Wait while Liferay pre-processes the .war files and places them in the `${liferay.home}websphere-deploy` directory.
3. Deploy the `.war` files generated in step 2 as modules in the Discovery Framework enterprise application. There are two ways to do this:
   - Through the Websphere Integrated Solutions Console.
   - At the command line, using `wsadmin`.

More details on each of these options appear in the following topics.

Deploying generated .war files on WAS 6.1 with the Integrated Solutions Console
You can use the IBM Integrated Solutions Console to deploy the `.war` files it finds in the `websphere-deploy` directory.

Important: The procedure in this topic does not work in versions of WAS prior to 6.1.0.9, due to a known problem in WebSphere Application Server (for details, see *http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg1PK36365*). If you are using a service pack prior to 6.1.0.9, use `wsadmin` to deploy the generated `.war` file, as documented in the next topic.

Note: These steps may need to be adjusted for alternate WAS configurations.

To manually deploy a generated `.war` file:

1. Go to **Applications** > **Enterprise Applications** and select the enterprise application created when you deployed the portal `.war` file. Click **Update**.
2. Select "Replace or add a single module."
3. Specify the path to deploy the file as the display name of the new module. For example, if you are adding `endeca-navigation-portlet.war`, specify the path as `endeca-navigation-portlet`.
4. Browse the remote file system to the newly created `.war` file in the Liferay deploy output directory. Continuing the example above, this might be `/home/endeca/liferay/websphere-deploy/endeca-navigation-portlet.war`. Click **Ok**.
5. Select the detailed install path and keep the defaults on all screens except the context root. Set the context root to match the display name of the new plugin (in this example, `/endeca-navigation-portlet/`).
6. Once it has successfully updated, click **Save directly to master configuration**.

Using wsadmin to deploy the generated .war file on WAS 6.1
You can also deploy the generated `.war` file at the command line using the `wsadmin` tool.

Note: These steps may need to be adjusted for alternate WAS configurations.

In the `wsadmin` tool, enter a command similar to the example below.

In this example, the enterprise application is named `DiscoveryFramework`. The module being added has the file name `endeca-navigation-portlet.war` and the display name `endeca-navigation-portlet`. This command is executed from the Liferay deploy output directory (that is, the directory containing the `endeca-navigation-portlet.war` file). In our example, this command is executed in `/home/endeca/liferay/websphere-deploy/.`.

```
[WAS]/AppServer/bin/wsadmin.sh -c "\$AdminApp update DiscoveryFramework
 modulefile {-operation addupdate -contents endeca-navigation-port¬
let.war -contextroot /endeca-navigation-portlet/ -contenturi endeca-nav¬
igation-portlet -usedefaultbindings}" –c "\$AdminConfig save"
```

Entering the license key on WAS 6.1

The Discovery Framework is build upon Liferay Portal Enterprise Edition. When the Discovery Framework is initially deployed and started, the first user to access the application is prompted for a license key.

To license your version of the Discovery Framework:

1. Download the license key from the Discovery Framework section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*. (For details on navigating to this section, see the topic "Downloading the Endeca Discovery Framework software for the full installation."
2. In the **Liferay Portal license** dialog box, enter the key in the **License Key** field.
3. Click **Save**. The portlet updates the dialog box with more detailed information about the license key.
4. Click **Save** again. The dialog box closes and you can proceed with your work.

Troubleshooting WAS 6.1 deployment

This topic discusses an issue to keep in mind when deploying the Discovery Framework on WAS 6.1.

Updating the Discovery Framework .war file

If you need to update the Discovery Framework `.war` file (not any individual plugin, but the portal `.war` itself), you must restart the WAS server. If you only restart the module, the restart might not be successful.

Installing the Discovery Framework on the WebSphere Application Server version 7

You can deploy the Discovery Framework as a standalone application on WebSphere Application Server (WAS) version 7.0.

Before following the steps here, consult the *Liferay Portal Administrator's Guide*, which contains portal deployment instructions and examples for WebSphere Application Server 7.0.

Note: The examples in this section are based on a Linux server WAS deployment. If you are installing on Windows, the steps will be similar, though you will need to substitute Windows executables and paths. In certain examples, backslashes are used to escape the dollar sign ($) character on Linux, because the shell would otherwise attempt a variable substitution for this character. These backslashes should not be required on a Windows system.

Note: Among the data sources in your Discovery Framework application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page. For details, see the topic "Specifying a default data source," located in the chapter "About data source configuration."

High-level overview of Websphere Application Server 7 deployment

This topic provides an overview of the steps you need to take to deploy the Discovery Framework on WAS 7.

Details on each of these steps appear in the topics that follow.

To deploy the Discovery Framework on WAS:

1. Deploy dependency `.jar` files. The exact list of required files and destination directories appears below.
2. Start (or restart) the WAS server.
3. Install the Discovery Framework `.war` file as an enterprise application.
4. Edit and deploy `portal-ext.properties`.
5. Create the `endeca-data-sources/*.json` data source configuration files.

   For more information, see the section "About data sources."
6. Install the Endeca theme, portlet components, and other framework `.war` files.
7. Start the Discovery Framework enterprise application, entering the license key upon first use.
8. Optionally, repeat step 6 for any additional plugins you want to add.

Deploying Discovery Framework dependency libraries on WAS 7

The Discovery Framework requires the deployment of several Java libraries.

These libraries are deployed to a global class loader, making them available to multiple applications.

To deploy the Discovery Framework dependency libraries:

1. Unzip the `.jar` files found in `endeca-portal-dependencies-<`*`version`*`>.zip`.
2. Upload the following list of `.jar` files from the `.zip` file to the WAS server's external library directory. (For example, if WAS is installed in `/usr/local/WAS/AppServer`, you would deploy the selected `.jar` files into `/usr/local/WAS/AppServer/lib/ext/`.)

```
annotations.jar
commons-lang.jar
cs_bindings.jar
cxf-2.2.8.jar
cxf-rt-databinding-jaxb-2.2.8.jar
endeca-images.jar
endeca-portal.jar
endeca_navigation.jar
ext-service.jar
geronimo-activation_1.1_spec-1.0.2.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-jaxws_2.1_spec-1.0.jar
geronimo-saaj_1.3_spec-1.0.1.jar
geronimo-stax-api_1.0_spec-1.0.1.jar
geronimo-ws-metadata_2.0_spec-1.1.2.jar
hsql.jar
jabsorb.jar
```

```
jackson-core-lgpl-1.6.2.jar
jackson-mapper-lgpl-1.6.1.jar
jaxb-api-2.1.jar
jaxb-impl-2.1.7.jar
jsr173_1.0_api.jar
log4j.jar
portal-kernel.jar
portal-service.jar
portlet.jar
slf4j-api.jar
slf4j-log4j12.jar
stax-1.2.0.jar
wsdl4j-1.6.2.jar
wstx.jar
XmlSchema-1.4.3.jar
```

3. Restart the WAS server so that it can pick up the newly available `.jar` files.

Extracting the standalone portal WAR on WAS 7

Before you can install the standalone portal WAR, you must extract it from its download package.

To extract the standalone portal WAR on WAS 7:

1. Unzip `endeca-portal-<version>.war.zip` into a temporary directory.
   This zip file contains the `endeca-portal-<version>.war` file and the `copyright.txt` file.
2. Read the `copyright.txt` file and then save it to the location of your choice.
3. Note the location of the `endeca-portal-<version>.war` file, as you will need to navigate to it in the next step.

Deploying the standalone portal WAR on WAS 7

After downloading and extracting the necessary files, you can deploy the Discovery Framework as an enterprise application in WebSphere Application Server, and then install portlets, themes, and other plugins as modules in that enterprise application.

The following steps document the installation procedure by using the IBM Integrated Solutions Console for a WebSphere Application Server installed and maintained without the use of the Deployment Manager, and consisting of one cell with one node and one server. The instructions may need to be adjusted for clustered environments, environments maintained with the Deployment Manager, or for environments where administration is performed by using tools like `wsadmin`, rather than the Integrated Solutions Console.

The following steps assume that no other applications are deployed in the same application server. If there are other applications, ensure that no applications are bound to context root `/` (or that any such applications are stopped during the Discovery Framework deployment). After following these steps, you will be able to adjust the context root for the Discovery Framework application, to ensure it does not conflict with other applications.

To deploy the Discovery Framework standalone portal WAR on WAS 7:

1. Start the WAS server.
2. Log in to the WAS Integrated Solutions Console, using the appropriate administrator credentials.
3. In the WAS Integrated Solutions Console, select **Applications** > **New Application** > **New Enterprise Application**.

4. Click to browse to and select the Endeca Discovery Framework WAR you downloaded earlier (`endeca-portal-<version>.war`), and then click **Next**.
5. Select **Choose to generate default bindings and mappings** and check the following options:
   - **Generate default bindings**
   - **Override existing bindings**
6. Still in the **Choose to generate default bindings and mappings** section, check **Use default virtual host name for Web and SIP modules**, and enter `default_host` in the text field. Click **Next**.
7. By default, the application name is **endeca-portal-<*version*>_war**. Set the application name to a more relevant name (for example, `DiscoveryFramework`). All other installation options can remain unchanged. Click **Next**.

   Note: Do not use spaces in the application name. For example, use **DiscoveryFramework** instead of **Discovery Framework**.

8. In **Map modules to servers**, accept the default settings and click **Next**.
9. In **Map context roots for Web modules**, set the context root to the desired path for your Discovery Framework installation, and then click **Next**.

   Note: Make a note of your context root, as you will need to reference it several times during the deployment process.

10. In **Install New Application**, confirm that your settings are correct and then click **Finish**.
11. Wait for installation and, if it is successful, click **Save directly to master configuration**.

Creating the Liferay Home directory on WAS 7

The remaining instructions in this section refer to a directory called Liferay Home. The Liferay Home directory is created relative to the user's home directory.

Manually create a Liferay Home directory (`/home/endeca/liferay/`), along with the following subdirectories:

- `/home/endeca/liferay/data`
- `/home/endeca/liferay/data/endeca-data-sources`
- `/home/endeca/liferay/websphere-deploy`

Editing the portal-ext.properties file for WAS 7 deployment

Before deploying your `portal-ext.properties` file, you must edit it.

Endeca's default version of `portal-ext.properties` is included in the package `endeca-portal-dependencies-<version>.zip`.

1. Open the `portal-ext.properties` file and add the following lines to the end of the file:

```
# Specify a directory where Liferay will "deploy" processed plugins.
# From this directory, WAS users will deploy WARs as modules in the
# Discovery Framework enterprise application.
#
auto.deploy.dest.dir=${liferay.home}/websphere-deploy
#
# Set this to true to enable JMX integration in
```

```
# com.liferay.portal.cache.EhcachePortalCacheManager.
#
ehcache.portal.cache.manager.jmx.enabled=false
```

Note: The destination directory (specified by the `auto.deploy.dest.dir` setting) must exist before the plugin is hot-deployed. In the above example, you must manually create the `websphere-deploy` directory if it does not exist.

2. Find the `portal.ctx` property at the beginning of `portal-ext.properties`. Change the value of this setting to be the same context root value you used when deploying the standalone portal WAR. However, do not include a trailing slash in the `portal.ctx` value.
   For example, use this value:

   ```
   portal.ctx=/mycompany/portal
   ```

   Do not use this value:

   ```
   portal.ctx=/mycompany/portal/
   ```

3. Save the file.

Configuring portal-ext.properties for WAS 7 deployment

After you edit your `portal-ext.properties` file, there are two ways to deploy it in WAS.

- By updating the application to include the `portal-ext.properties` file.
- By uploading the `portal-ext.properties` file to the Liferay Home directory on the server.

Both methods are described in the following topics.

Updating the application to include the portal-ext.properties file on WAS 7

After you create the `portal-ext.properties` file, you can use the IBM Integrated Solutions Console to update the portal WAR module with the additional file.

This topic documents the use of the Integrated Solutions Console to update the Discovery Framework application, to include `portal-ext.properties` in the `endeca-portal.war` module. These steps may be performed with the `wsadmin` tool instead of the Integrated Solutions Console and may need to be adjusted for alternate WAS configurations.

Note: In order to make changes to the `portal-ext.properties` file, users will need to repeat these steps to update the application with updated versions of the `portal-ext.properties` file. In some environments, it may be more appropriate to deploy the `portal-ext.properties` file to the Liferay Home directory, where it can be updated without updating the deployed application. That option is described in the next topic.

To deploy a `portal-ext.properties` file in the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications** and select the enterprise application created when you deployed the portal WAR. Click Update.
2. Select **Replace or add a single file**.
3. Specify the path to deploy the file into the `WEB-INF/classes` directory of the portal Web application. For example: `endeca-portal-<version>.war/WEB-INF/classes/portal-ext.properties`
4. Browse to where you created the file on your computer.
5. Once the file has successfully updated, click **Save directly to master configuration**.

Uploading portal-ext.properties to Liferay Home on the server on WAS 7
After you create the `portal-ext.properties` file, you can manually upload it to WAS.

To manually upload the `portal-ext.properties` file:

Upload the `portal-ext.properties` file to the Liferay Home directory. For example: `/home/endeca/liferay/portal-ext.properties`.
Liferay reads these properties when the Discovery Framework application is started.

Example settings for portal-ext.properties on WAS 7
Endeca's default version of `portal-ext.properties` is included in the package `endeca-portal-dependencies-<version>.zip`.

This file serves as a useful starting point for configuration of the portal properties, and should be deployed to the application server according to the steps described in a previous topic.

Note: Most of the settings in the default `portal-ext.properties` file are not specific to deployment on WAS 7. However, the following additional settings (which you must add to the file as described in the topic "Editing the portal-ext.properties file for WAS 7 deployment") are important for portlet deployment on WAS:

```
auto.deploy.dest.dir=/home/endeca/liferay/websphere-deploy
ehcache.portal.cache.manager.jmx.enabled=false
```

Deploying Endeca data source configuration on WAS 7

To configure one or more MDEX Engines as data sources for the Discovery Framework, a JSON configuration file needs to be deployed for each MDEX Engine.

These files should be deployed relative to the Liferay Home directory. Sample data source configuration files are provided as `.json.sample` files in the `endeca-portal-dependencies-<version>.zip` file you downloaded.

To deploy Endeca data source configuration:

Upload the files to the `data/endeca-data-sources/` subdirectory.
For example: `/home/endeca/liferay/data/endeca-data-sources/default.json`

Starting the application on WAS 7

Once the Discovery Framework application has been deployed, and the `portal-ext.properties` file has been configured and deployed, the application needs to be started.

The following steps describe this process in the IBM Integrated Solutions Console.

To start the application:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications** and select the enterprise application created when you deployed the portal WAR.
2. If it is not already running, click **Start** to start it.
3. View your deployed application at the root context of the server.
4. Restart WAS 7.

Deploying components and other plugins in WAS 7

This section describes how to deploy components, themes, hooks, and other plugins in WAS 7.

These plugins are located in the `components-<version>.zip` package.

About Liferay component pre-processing in WAS 7
WAS does not support the hot deployment of components. However, Liferay's deployment code must update plugins by adding necessary libraries and configuration files.

For example, Liferay's portlet deployment code adds the following important piece of configuration to a portlet component's `web.xml` file:

```
<context-param>
  <param-name>com.ibm.websphere.portletcontainer.PortletDeploymentEn¬
abled</param-name>
  <param-value>false</param-value>
</context-param>
```

This context parameter is important for WAS deployment, as it ensures that WAS's portal server does not attempt to load the new portlet, and instead allows the Discovery Framework to load the newly deployed portlet.

For this reason, Liferay must be allowed to pre-process components before they are deployed to WAS. You upload your `.war` files to Liferay's `deploy` directory so that Liferay's deployer can find and process them.

Deploying components in WAS 7
Liferay must pre-process Discovery Framework components before they can be deployed in WAS 7.

Important: The Discovery Framework requires the Endeca Theme in order to start up and allow you to enter a license key. Even if you do not intend to use the Endeca Theme in production, you should deploy the Endeca Theme (`endeca-theme-<version>.war`).

To deploy Discovery Framework components in WAS 7:

1. Copy all component `.war` files to `${liferay.home}/deploy`.
2. Wait while Liferay pre-processes the .war files and places them in the `${liferay.home}websphere-deploy` directory.
3. Deploy the `.war` files generated in step 2 as modules in the Discovery Framework enterprise application. There are two ways to do this:
   - Through the Websphere Integrated Solutions Console.
   - At the command line, using `wsadmin`.

More details on each of these options appear in the following topics.

Deploying generated .war files on WAS 7 with the Integrated Solutions Console
You can use the IBM Integrated Solutions Console to deploy the `.war` files it finds in the `websphere-deploy` directory.

Note: These steps may need to be adjusted for alternate WAS configurations.

To deploy a generated `.war` file with the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications** and select the enterprise application created when you deployed the portal `.war` file. Click **Update**.
2. Select "Replace or add a single module."

3. Specify the path to deploy the file as the display name of the new module. For example, if you are adding `endeca-navigation-portlet.war`, specify the path as `endeca-navigation-portlet`.
4. Browse the remote file system to the newly created `.war` file in the Liferay deploy output directory. Continuing the example above, this might be `/home/endeca/liferay/websphere-deploy/endeca-navigation-portlet.war`. Click **Ok**.
5. Select the detailed install path and keep the defaults on all screens except the context root. Set the context root to match the display name of the new plugin (in this example, `/endeca-navigation-portlet/`).
6. Once it has successfully updated, click **Save directly to master configuration**.

Using wsadmin to deploy the generated .war file on WAS 7
You can also deploy the generated `.war` file at the command line using the `wsadmin` tool.

Note: These steps may need to be adjusted for alternate WAS configurations.

In the `wsadmin` tool, enter a command similar to the example below, where the command is executed from the Liferay deploy output directory (that is, the directory containing the `endeca-navigation-portlet.war` file):

```
[WAS]/AppServer/bin/wsadmin.sh -c "\$AdminApp update DiscoveryFramework
modulefile {-operation addupdate -contents endeca-navigation-port¬
let.war -contextroot /endeca-navigation-portlet/ -contenturi endeca-naviga¬
tion-portlet -usedefaultbindings}" –c "\$AdminConfig save"
```

In this example:

- The enterprise application is named `DiscoveryFramework`.
- The module being added has the file name `endeca-navigation-portlet.war` and the display name `endeca-navigation-portlet`.
- The command is executed in `/home/endeca/liferay/websphere-deploy/.`.

Entering the license key on WAS 7

The Discovery Framework is build upon Liferay Portal Enterprise Edition. When the Discovery Framework is initially deployed and started, the first user to access the application is prompted for a license key.

To license your version of the Discovery Framework:

1. Download the license key from the Discovery Framework section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*. (For details on navigating to this section, see the topic "Downloading the Endeca Discovery Framework software for the full installation.")
2. In the **Liferay Portal license** dialog box, enter the key in the **License Key** field.
3. Click **Save**. The portlet updates the dialog box with more detailed information about the license key.
4. Click **Save** again. The dialog box closes and you can proceed with your work.

Troubleshooting WAS 7 deployment

This topic discusses an issue to keep in mind when deploying the Discovery Framework on WAS 7.

Updating the Discovery Framework .war file

If you need to update the Discovery Framework `.war` file (not any individual plugin, but the portal `.war` itself), you must restart the WAS server. If you only restart the module, the restart might not be successful.

Discovery Framework interaction with IAP 6.1.4

The Discovery Framework works with versions 6.1.3 and 6.1.4 of the MDEX Engine, but must be configured to take advantage of the new features in version 6.1.4.

 **Note:** All MDEX Engines backing a specific installation should be either 6.1.3 or 6.1.4.

Configuring the Discovery Framework on Tomcat to work with IAP 6.1.4

This topic describes how to configure a Discovery Framework implementation running on Tomcat to work with IAP 6.1.4.

To configure the Discovery Framework on Tomcat to work with IAP 6.1.4:

Copy the 6.1.4 `endeca_navigation.jar` file to the `tomcat-<version>/lib/ext` directory.
This overwrites the existing version of `endeca_navigation.jar`.

All 6.1.4 MDEX Engine features automatically work in the Discovery Framework. However, there is no built-in support for the 6.1.4 Presentation API features. To expose the new features, you have to add your own `QueryFunction` using the `QueryFunction` extension point, or manipulate the `ENEQuery` directly within the component.

Configuring the Discovery Framework on WAS to work with IAP 6.1.4

This topic describes how to configure a Discovery Framework implementation running on Websphere Application Server to work with IAP 6.1.4.

To configure the Discovery Framework on Tomcat to work with IAP 6.1.4:

Copy the 6.1.4 `endeca_navigation.jar` file to the `WAS/AppServer/lib/ext` directory.
This overwrites the existing version of `endeca_navigation.jar`.

All 6.1.4 MDEX Engine features automatically work in the Discovery Framework. However, there is no built-in support for the 6.1.4 Presentation API features. To expose the new features, you have to add your own `QueryFunction` using the `QueryFunction` extension point, or manipulate the `ENEQuery` directly within the component.

Installing the components-only package

From time to time you may want to install a components-only update to the Discovery Framework. In such instances, it is not necessary to re-install the entire Discovery Framework.

This section contains instructions for component-only installations on any of the supported application servers.

Important: Because application servers may cache JavaScript and CSS, the cache may need to be cleared or the server restarted after installation in order for component upgrades to take effect.

Downloading the Endeca Discovery Framework software for a component installation

You can download the Endeca Discovery Framework from the Downloads section of the Endeca Developer Network (EDeN).

To download the Discovery Framework software:

1. If you have not previously done so, establish a Support account with download access through the Support section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*. This enables the Endeca Support and Customer Care groups to track which versions of the software you are using.
2. Navigate through the EDeN site as follows:
   a) On the EDeN homepage, click **Downloads**.
   b) On the **Tools and Utilities** page, find the **Product Downloads** section and click **View and download purchased products**.
   c) On the **Product Downloads** page, find and click **Discovery Framework**.
   d) In the **Current Releases** table, click **Discovery Framework *<version>***.

   The **Product Download** page contains links to all available Discovery Framework packages.
3. Download the following Discovery Framework zip files:
   - `components-dependencies-<version>.zip`
   - `components-<version>.zip`.
   - `corda-<version>.zip`

Prerequisites to installing a Discovery Framework component release

Due to a number of product enhancements in this release, you need to complete the following tasks before you install a components-only package.

To prepare your Discovery Framework portal for a component-only release:

1. Download `corda-<version>.zip` and re-install Corda, following the instructions found in the section "Installing Corda."
2. Download `components-<version>-dependencies.zip` and redeploy the `endeca-portal.jar` file, following the instructions for your application server:
   - For Tomcat 5.5, deploy `endeca-portal.jar` to `endeca-portal/apache-tomcat-5.5.x/common/lib/ext`.
   - For the default Tomcat 6 bundle, WAS 7, or WAS 6.1, deploy `endeca-portal.jar` to the`/lib/ext` directory under your application server home directory.
3. In the `portal-ext.properties` file (located in your `endeca-portal` directory on Tomcat, or your `{liferay.home}` directory on WAS), add the following line:

```
portlet.add.default.resource.check.whitelist=58,86,87,88,103,113,145,en¬
decaresultsexportportlet_WAR_endecaresultsexportportlet
```

4. Restart your server.
   Once these steps have completed successfully, you are ready to install the component-only update.

Installing the component-only package on any supported version of Tomcat

This topic describes how to install a component upgrade when you are running the Discovery Framework on a Tomcat application server.

 **Note:** No backing up of data or removal of existing components is required.

To install the Discovery Framework components-only package on a supported version of Tomcat:

1. If you have not already done so, download `components-<version>.zip`, as described at the beginning of this section.
2. Extract `components-<version>.zip` to your `endeca-portal\deploy` directory, overwriting existing component versions.
3. Resume using the Discovery Framework.
   No restart is required.

Installing the components-only package on WebSphere

This topic describes how to install a component upgrade when you are running the Discovery Framework on WebSphere Application Server (WAS) version 7.

To install the Discovery Framework components-only package on WAS:

1. Download `components-<version>.zip`, as described at the beginning of this section.
2. Extract `components-<version>.zip` to your `${liferay.home}/deploy` directory, overwriting existing component versions.

    **Note:** For information about the Liferay home directory, see the topic, "Creating the Liferay home directory," which appears earlier in this chapter.

3. Do one of the following:
   - Manually deploy the generated `.war` files.
   - Use wsadmin to deploy the generated `.war` files.

   Both of these procedures are described in the following topics.

Deploying generated .war files on WAS 7 with the Integrated Solutions Console

You can use the IBM Integrated Solutions Console to deploy the `.war` files it finds in the `websphere-deploy` directory.

 **Note:** These steps may need to be adjusted for alternate WAS configurations.

To deploy a generated `.war` file with the Integrated Solutions Console:

1. Go to **Applications** > **Application Types** > **WebSphere Enterprise Applications** and select the enterprise application created when you deployed the portal `.war` file. Click **Update**.
2. Select "Replace or add a single module."
3. Specify the path to deploy the file as the display name of the new module. For example, if you are adding `endeca-navigation-portlet.war`, specify the path as `endeca-navigation-portlet`.
4. Browse the remote file system to the newly created `.war` file in the Liferay deploy output directory. Continuing the example above, this might be `/home/endeca/liferay/websphere-deploy/endeca-navigation-portlet.war`. Click **Ok**.
5. Select the detailed install path and keep the defaults on all screens except the context root. Set the context root to match the display name of the new plugin (in this example, `/endeca-navigation-portlet/`).
6. Once it has successfully updated, click **Save directly to master configuration**.

Using wsadmin to deploy the generated .war file on WAS 7

You can also deploy the generated `.war` file at the command line using the `wsadmin` tool.

Note: These steps may need to be adjusted for alternate WAS configurations.

In the `wsadmin` tool, enter a command similar to the example below, where the command is executed from the Liferay deploy output directory (that is, the directory containing the `endeca-navigation-portlet.war` file):

```
[WAS]/AppServer/bin/wsadmin.sh -c "\$AdminApp update DiscoveryFramework
modulefile {-operation addupdate -contents endeca-navigation-port¬
let.war -contextroot /endeca-navigation-portlet/ -contenturi endeca-naviga¬
tion-portlet -usedefaultbindings}" –c "\$AdminConfig save"
```

In this example:

- The enterprise application is named `DiscoveryFramework`.
- The module being added has the file name `endeca-navigation-portlet.war` and the display name `endeca-navigation-portlet`.
- The command is executed in `/home/endeca/liferay/websphere-deploy/.`.

Downloading the Endeca Discovery Framework documentation

You can download documentation for the Endeca Discovery Framework from the Downloads section of the Endeca Developer Network (EDeN).

To download Discovery Framework documentation:

1. If you have not previously done so, establish a Support account with download access through the Support section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*. This enables the Endeca Support and Customer Care groups to track which versions of the software you are using.
2. Navigate through the EDeN site as follows:
   a) On the EDeN homepage, click **Downloads**.

b) On the **Tools and Utilities** page, find the **Product Downloads** section and click **View and download purchased products**.
c) On the **Product Downloads** page, find and click **Discovery Framework**.
d) In the **Current Releases** table, click **Discovery Framework <*version*>**.

The **Product Download** page contains links to all available Discovery Framework packages.

3. Download the **Discovery Framework Documentation** package to your desired location.

Note: The Discovery Framework Documentation package is distributed as a zip file.

4. Extract the Discovery Framework Documentation zip file using a decompression utility. Optionally, you can extract the files to the `endeca-portal\doc` directory that was created when you installed the Discovery Framework.

Note: You can also view a searchable version of the Discovery Framework documentation online in the EDeN Knowledge Base.

Chapter 3

About data source configuration

Data sources and query states are how the Discovery Framework represents connections to MDEX Engines and queries to those MDEX Engines. This section discusses how to create and configure data sources, and outlines the default interaction model between related data sources.

About data sources

Every instance of a component that needs to query the MDEX Engine is backed by a particular data source. That data source, which represents a pointer to a specific MDEX Engine, is used to maintain application state for each user's session.

The different components in your Discovery Framework application can connect to different data sources.

Note: Most administrative components do not require a backing data source.

Data source syntax

Data source files are written in the JSON language.

Every data source must include `server` and `port` keys. All other settings are optional. If `name` and `id` are not included, they are both given the name of the file defining the data source by default. For example, a file named `parts.json` results in a data source with the `name` and `id` "parts." The `id` cannot contain spaces. Therefore, if you do not specify an explicit id, make sure that your file name does not contain any spaces.

The `name` setting is normally the only user-visible identification of a data source. The `id` setting is only used internally, and the `description` setting is only used for logging and debugging.

Note: SSL configuration file paths are relative to the directory containing the JSON data source files. This is typically, but not always, the `endeca-portal\data\endeca-data-sources` directory.

Important: If you have access to MDEX 7 Early Access edition and want to connect to an MDEX 7 data source, see the section "Discovery Framework interaction with MDEX 7 Early Access edition" for important additional information.

Data source example

The following example shows a JSON data source file based on wine merchant data.

```
{
 "server":"server01.lab.acme.com",
 "port":"15000",
 "apiVersion" : "ENE_QUERY",
 "sslConfig": {
  "caFile":"truststore.ks",
  "caPassword":"endeca",
  "certFile":"keystore.ks",
  "certPassword":"endeca"
 }
 "id":"ds-id",
 "name":"Descriptive DataSource name",
 "description":"Detailed information about this DataSource",
 "baseFunctions": [
  {
   "class":"com.endeca.portal.data.functions.RecordFilter"
   "recordFilter":"Regions:Midwest"
  },
  {
   "class":"com.endeca.portal.data.functions.RangeFilter"
   "property":"P_Price",
   "rangeOperator":"GTEQ",
   "value1": "25"
  },
  {
   "class":"com.endeca.portal.data.functions.RefinementFilter"
   "dimValId":"123",
   "dimensionId":"121"
  }
        ],
"securityEnabled": "true",
"inheritSecurity": "true",
"securityFilters": {
     "frenchFilter": {
           "class":"com.endeca.portal.data.functions.RecordFilter",
           "recordFilter":"OR(Region:Bordeaux,Region:Burgundy)"
     }
},
"rolePermissions": {
     "French Wine": ["frenchFilter"]
}
}
```

Connecting to an MDEX 7 data source

In order to connect to an MDEX 7 data source, the JSON data source file requires an additional attribute.

The setting, `apiVersion`, has two possible values: `ENE_QUERY`, used for MDEX 6 data sources, and `DISCOVERY_SERVICE`, used for MDEX 7 data sources. If the setting is not present, its value defaults to `ENE_QUERY`.

The following is an example `default.json` file establishing a connection to an MDEX 7 data source:

```
{
 "server":"server01.lab.acme.com",
 "port":"15000",
```

```
 "apiVersion" : "DISCOVERY_SERVICE"
}
```

Important: MDEX 7 features are in an Early Access state. The interfaces and behavior of these Early Access features may change in later releases of MDEX 7 or of the Discovery Framework, based on information gathered during the Early Access period. These capabilities are not supported for use in production.

MDEX 7 data source limitation

Parent/child data sources are not supported on MDEX 7. You should not include the `parentId` attribute in your MDEX 7 data source JSON configuration files.

About the sample MDEX Engine data sources

The Discovery Framework ships with sample data sources. You can use the samples as a model for your own data sources

The following sample data sources are located in the `endeca-portal\data\endeca-data-sources` directory:

- `base-filters-data-source.json.sample`
- `child-data-source.json.sample`
- `secured-data-source.json.sample`
- `simple-data-source.json.sample`
- `ssl-enabled-data-source.json.sample`

This directory also contains a `default.json` file, containing host and port information only. The default data source is discussed later in this section.

Specifying a default data source

Among the data sources in your Discovery Framework application, you must always include a default data source. This data source is automatically assigned to all data-source-backed components when they are initially added to a page.

When you initially start the Discovery Framework, the `id` of this data source is "default." While this setting remains unchanged, you must include a data source with the `id` “default." Upon installation, a `default.json` data source file (with an implicit `id` of "default") is included in the `endeca-portal\data\endeca-data-sources` directory.

Alternatively, you can change the default data source name to that of another data source file. To do so:

1. In the **Liferay Control Panel**, go to the **Discovery Framework Settings** page.
2. In the **Discovery Framework Settings** page, change the `df.defaultDataSource` setting to a different value.
3. Restart the **Discovery Framework**.
4. Make sure you include a data source with that new value as its `id`.

Note: For more information about **Discovery Framework Settings**, see the *Discovery Framework Installation Guide*.

Adding data sources to the Discovery Framework

It is possible to add new data sources to the Discovery Framework. Your Endeca components can then access the data sources you have added.

To add a data source to the Discovery Framework:

1. Create a new JSON file in `endeca-portal\data\endeca-data-sources`.
   For definition examples, see the sample data sources located in the same directory.
2. After creating the new file on disk, do one of the following:
   - Restart the Discovery Framework.
   - In the Control Panel, select **Data Sources**, and then click **Update data sources**.

Note: If your data source does not appear after completing step 2, it probably means that your data source definition file contains invalid JSON syntax. You can confirm this by looking for a message about invalid syntax in the Discovery Framework log `df.log`. Check the log, edit your syntax, and try the steps above again.

Changing an Endeca component's data source

If more than one data source has been configured for the application, the data source for an individual portlet instance can be changed.

To change the data source for an Endeca component that can be bound to a data source:

1. In the header of the component whose data source you want to change, select the `...` icon, and then select **Preferences**.
2. Select the new data source in the drop down menu, and click **Update data source**.
   You should see that the component has been successfully bound to a new data source.
3. Click **Return to Full Page**.

Note: This procedure only changes the data source for that single instance of the component.

Configuring aggregated records in a data source

Aggregated records allow you to group records by dimension or property values.

By configuring aggregated records, you enable the MDEX Engine to handle a group of multiple records as though it were a single record, based on the value of the rollup key. A rollup key can be any property or dimension that has its rollup attribute enabled.

You can use parent-child data source relationships to create multiple data sources with different aggregations. The parent data source can use base records, while multiple child data sources each aggregate by a different rollup key.

For details on how individual components handles record aggregation, see the *Discovery Framework Component Catalog*.

Note: The aggregated record feature is only supported on MDEX 6 data sources. It is not supported on MDEX 7 data sources.

Note: Aggregated records in the Discovery Framework are subject to the same constraints as they are in the MDEX Engine: you cannot control the order of base records within an aggregate, and you cannot control which record is used as the representative record (beyond controlling overall sort order).

Aggregated record configuration

To configure a data source for aggregated records, you include the `RecordAggregator QueryFunction` class in the data source's JSON configuration file. The `RecordAggregator` contains two configuration properties: `rollupKey` and `aggCount`.

The `rollupKey` property is used to specify the dimension or property by which records in a navigation object's record list should be aggregated.

The `aggCount` property is used to limit the number of base records returned with the aggregated record. There are three possible values for `aggCount`:

- `ZERO_ERECS_PER_AGGR`: none of the records that compose the aggregated record will be returned. (In most cases, `ZERO_ERECS_PER_AGGR` is not useful. Because the MDEX Engine returns zero records, your components have almost no data to work with.)
- `ONE_EREC_PER_AGGR`: one of the records that composes the aggregated record will be returned.
- `ALL_ERECS_PER_AGGR` : all of the records that compose the aggregated record will be returned. (Due to performance implications, use of `ALL_ERECS_PER_AGGR` is discouraged.)

If no `aggCount` value is set, the default is `ONE_EREC_PER_AGGR`.

Aggregated record example

In the following example data source configuration file, the **P_Winery** property is the rollup key, and one of the records that composes the aggregated record will be returned.

```
{
 "server":"server01.lab.acme.com",
 "port":"15000",
  "baseFunctions": [
  {"class":"com.endeca.portal.data.functions.RecordAggregator",
   "rollupKey":"P_Winery",
   "aggCount":"ONE_EREC_PER_AGGR"
  }
 ]
}
```

Configuring the Discovery Framework to connect to a secured MDEX Engine

This topic provides a high-level description of how you can set up the Discovery Framework to connect to a secured (HTTPS) MDEX Engine.

Several of the steps below refer to chapter 3 of the *Endeca Platform Services Security Guide*, entitled "Using Endeca SSL Certificate Utilities." The *Endeca Platform Services Security Guide* is available as part of the Platform Services documentation set on EDeN. Before attempting these steps, make sure you have a copy of that guide at hand.

Note: The steps below assume you are using the Discovery Framework Tomcat bundle.

1. Generate the SSL certificate files for the Dgraph using the `enecerts` utility. This utility is available in both the Platform Services and MDEX Engine installation. For instructions, refer to the section "Generating SSL certificates" in the *Endeca Platform Services Security Guide*.
2. Generate the Java KeyStore (JKS) files using the `endeca-key-importer.jar` utility from the Endeca Platform Services installation. For instructions, refer to the topic "Converting PEM-format keys to JKS format" in chapter 3 of the *Endeca Platform Services Security Guide*.
3. Place the JKS keys into the directory containing the JSON data source files. This is typically, but not always, the `endeca-portal\data\endeca-data-sources` directory.
4. Specify the `caFile`, `certFile`, `caPassword`, and `certPassword` in the appropriate JSON data source file.
   The following example is extracted from a larger data source file:

```
"sslConfig": {
  "caFile":"truststore.ks",
  "caPassword":"endeca",
  "certFile":"keystore.ks",
  "certPassword":"endeca"
 }
```

5. Restart the Discovery Framework.

Data source role-based security

The default `MDEXSecurityManager` implementation supports the configuration of filters associated with Liferay roles that have been assigned to a user's login.

The supported filters are as follows:

- **securityEnabled:** enabling/disabling security filters on all queries issued to this data source.
- **securityFilters:** definition of all security filters to be used by this data source (any extension of `QueryFilter`).

  Note: Record filters are the only supported type of `securityFilter`.

- **rolePermissions:** role mappings to the security filters that have been defined for the data source.
- **inheritSecurity:** enabling/disabling of security filter inheritance, based on data source relationships defined via the `parentDataSource` property.
- **parentDataSource:** when `inheritSecurity` is true, this property is used to find all ancestors of each data source being secured. For each data source, the list of security filters to be applied is a combination of all security filters configured for every ancestor data source, plus any found with the data source configuration itself.

Note: For more information on security, see the chapter "Security extensions to the Discovery Framework" in the *Discovery Framework Extension Guide*.

About data source relationships

This topic describes the relationship between parent and child data sources.

Important: The way data sources interact with each other can vary based on the Data Source State Manager your portal is using. The information below only applies to the default implementation of the State Manager provided with the released version of the Discovery Framework.

Discovery Framework data sources can have parents and children. These relationships can be set in a data source's JSON file through the `parentDataSource:parent-ds-id` property.

In behavior, child data sources act like immutable filter sets. If you are using a child data source, the `baseFunctions` setting in the data source file should be specified with some set of filters. When a component connected to a child data source attempts to get the current query state, the query state is the query state of the data source's parent with all of the static filters on the child (as specified in configuration as `baseFunctions`) appended to it.

When a component connected to a child data source attempts to change that data source's query state (by setting a refinement from Guided Navigation, for example), the change actually happens on the parent data source. Such an operation keeps applying to a data source's parent until it finds a data source without a parent. When any data source's query state is changed, the query states of all of its children are effectively changed as well, and any components connected to them will refresh.

Note: Role-based security filters can be inherited by descendant data sources when using the Security Manager, via the `parentDataSource` property.

MDEX 7 data source limitation

Parent/child data sources are not supported on MDEX 7. You should not include the `parentId` attribute in your MDEX 7 data source JSON configuration files.

Provided QueryFunction classes in the Discovery Framework

This topic defines the `QueryFunction` classes included in the Discovery Framework.

Note: As noted in the table below, some `QueryFunction` classes behave differently in MDEX 6 and MDEX 7. For more information about MDEX 7, see the section "Discovery Framework interaction with MDEX 7 Early Access edition" in the *Discovery Framework Extension Guide*.

Filters

Function Class	ENEQuery Equivalent	Configuration Properties	Notes
`EQLFilter`	`setNavRecord¬ StructureExpr()`	`eqlFilter:` String	Applying an EQL filter always overwrites previous `EQLFilters`. .

Function Class	ENEQuery Equivalent	Configuration Properties	Notes
NegativeRefinement¬Filter	setNavRecordFilter()	dimValName: String dimValId: String dimensionName: String dimensionId: String ancestors: String[]	
RangeFilter	setNavRangeFilters()	property: String rangeOperator: (LT|LTEQ|GT|GTEQ¬|BTWN|GCLT|GCGT|GCBTWN) value1: numeric value2: numeric (optional) value3: numeric (optional)	Only available in IAP 6.
RecordAggregator	setNavRollupKey()	rollupKey: String aggCount: ({{ONE_EREC_PER_AGGR|ALL_ERECS_PER_AGGR|ZERO_ERECS_PER_AGGR)}}	Only available in IAP 6.
RecordFilter	setNavRecordFilter()	recordFilter: String	
RecordSpecListFilter	setERecs()	recSpecs: String[]	Only available in IAP 6. Applying a RecordSpecListFilter always overwrites previous RecordSpecListFilters.
RefinementFilter	setNavDescriptors()	dimValId: long dimensionId: long multiSelect: (AND|OR|NONE) (optional)	

Function Class	ENEQuery Equivalent	Configuration Properties	Notes
		navigable: (true|false) (optional)	
SearchFilter	setNavERecSearches()	searchInterface: String terms: String matchMode: (ALL|PARTIAL|ANY|ALLANY|ALLPARTIAL|PARTIALMAX|BOOLEAN)	

Configuration functions

Function Class	ENEQuery Equivalent	Configuration Properties	Notes
AnalyticsQueryConfig	setAnalyticsQuery()	analyticsQuery: String	
BreadcrumbsConfig	N/A	returnFullPath: boolean (optional)	MDEX 7 Discovery Service only. Has no effect in IAP 6.
DimensionSearch¬Config	• setDimSearchTerms() • setDimSearch¬NavDescriptors() • setDimSearch¬NavRangeFilters() • setDimSearch¬NavRecordFilters() • setDimSearch¬NumDimValues() • setDimSearch¬NavRecord¬StructureExpr()	searchTerm: String maxPerDimension: int (optional) dimensionId: String (optional) dimValId: String (optional) matchMode: (ALL|PARTIAL|ANY|ALLANY|ALLPARTIAL|PARTIALMAX|BOOLEAN) relevanceRanking¬Strategy: String (optional)	
ExposeRefinement	setNavExposed¬Refinements()	dimValId: String dimensionId: String	At least one of dimValId or dimensionId

Function Class	ENEQuery Equivalent	Configuration Properties	Notes
		`ownerId`: String (optional) `dimExposed`: boolean (optional) `exposeAll`: boolean (optional) `maxRefinements`: int (optional)	is required. `ownerId` can be the ID of a NavConfig instance. `dimExposed` indicates whether the dimension is exposed. `exposeAll` indicates whether the dimension is fully exposed (that is, the "More..." link is selected). `maxRefinements` is only valid in the MDEX 7 Discovery Service. It has no effect in IAP 6. The default value is 100.
`NavConfig`	`setNavAllRefinements`	`exposeAllRefinements`: boolean	

`ResultsConfig`	`setNavNumERecs()`
	`setNavERecsOffset()`
	`setSelection`
		`recordsPerPage`: int
		`offset`: int (optional)
		`columns`: String[] (optional)
		`numBulkRecords`: int (optional)	

`ResultsSummaryConfig`	N/A		
`SearchKeysConfig`	N/A		MDEX 7 Discovery Service only. Has no effect in IAP 6.
`SortConfig`	`setNavActiveSortKeys()`	`property`: String	

Function Class	ENEQuery Equivalent	Configuration Properties	Notes
		`ascending:` boolean	

Implementing new QueryFunction classes

In addition to using the provided `QueryFunction` classes, you can also create your own.

Endeca provides tools you can use to create your own `QueryFilter` and `QueryConfig` classes. These classes define new operations that can be applied to a `QueryState`. Both of these classes inherit from `com.endeca.portal.data.QueryFunction`. A `QueryFilter` reduces the result set, while a `QueryConfig` modifies the configuration of the results returned. This includes functions such as paging, sorting, exposing refinements, and setting the number of results returned.

`QueryFilters` are persisted to the data source state when a component calls `setQueryState()` and are therefore shared across all components that are bound to the data source. `QueryConfigs` are not persisted and therefore are specific to the component that applied them.

Related Links

Creating a QueryFunction class on page 55
This topic describes the steps required to create a new `QueryFunction` class.

Implementing the QueryFunction class on page 56
This topic describes the steps needed to implement your new `QueryFunction` class.

Using your custom QueryFunction on page 56
In order to use your new `QueryFunction`, you must deploy it to the Discovery Framework.

Adding the new .jar file to your Eclipse build path on page 57
If you are using Eclipse as your IDE, you need to add the new `.jar` file to your build path of your custom component.

Creating a QueryFunction class

This topic describes the steps required to create a new `QueryFunction` class.

The steps below create a `QueryFilter` class, but the steps are analogous for creating a `QueryConfig` class.

Note: In order to create `QueryFunction` classes, you must install the Component SDK, which is a separate download.

To create a new `QueryFilter` class:

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-queryfilter.bat` *`your-query-filter-name`*
   - On Linux: `./create-queryfilter.sh` *`your-query-filter-name`*

This command creates a `<your-query-filter-name>-filter` directory under `endeca-extensions`. This directory is an Eclipse project that can be imported directly into Eclipse if that is your IDE.

The `endeca-extensions` directory also contains an empty sample implementation of either a `QueryFilter` or a `QueryConfig`, depending on which batch script you ran. This has no effect on `QueryState` in its original form. The skeleton implementation creates source files that do the following:

- Extends either `QueryFilter` or `QueryConfig`.
- Creates stubs for the abstract methods you need to implement: `applyToENEQuery`, `applyToDiscoveryServiceQuery`, and `toString`.
- Creates default implementations for `getSetters` and `getGetters`. These use static `setters` and `getters` member properties that use reflection to extract the appropriate methods from the class.
- Creates a no-arg, protected, empty constructor. (The protected access modifier is optional, but recommended.)
- Creates a private member variable for logging.

Implementing the QueryFunction class

This topic describes the steps needed to implement your new `QueryFunction` class.

To implement your new `QueryFunction`, you must:

- Add private filter or configuration properties.
- Create getters and setters for any filter properties you add.
- For any property that is not a String, create a setter property that takes a String and does conversion.
- Define a no-arg constructor (protected access modifier optional, but recommended).
- Implement the abstract methods `getSetters`, `getGetters`, `applyToENEQuery`, `applyToDiscoveryServiceQuery`, and `toString`. You can use the `getSetters` and `getGetters` methods from the sample `QueryFunction`.

Note: Because `.toString()` is used in `.equals()`, you should make sure that two `QueryFunction` objects that are the same return the same value. Specifically, `.toJSON().toString()` does not guarantee ordering of JSON properties, so two `QueryFunction` objects with the same member values may not return the same value if `.toString()` was implemented using `.toJSON().toString`.

Using your custom QueryFunction

In order to use your new `QueryFunction`, you must deploy it to the Discovery Framework.

The *`your-query-filter-name`*`-filter|config` directory that you created 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.

Restart the Discovery Framework so that the portal picks up the new class file.

Once you have deployed your custom `QueryFunction`, you can use it in any component.

Adding the new .jar file to your Eclipse build path

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

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

1. Right-click on the project, and select **Build Path** > **Configure Build Path**.
2. Click the **Libraries** tab.
3. Click **Add Variable**, select **DF_GLOBAL_LIB** (which you should have added while setting up the SDK), and then click **Extend**.

   For more information, see the topic "Configuring Eclipse for Component Development" in the section "Installing and using the Component SDK."
4. Open the `ext/` directory and select the `.jar` file containing your custom `QueryFunction`.
5. Click **OK**.

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

Obtaining data source results

The `ENEQueryResults` class from the Presentation API is used to represent results of queries.

Components are always encouraged to add the relevant `QueryConfig` to specify what types of results they need. Calls to `DataSource.execute()`, without any arguments, will continue to work on ENE Presentation API data sources, but are deprecated.:

```
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.

```
ENEQueryResults eneResults = results.getENEQueryResults();
```

You can also make other local modifications to your query state before executing by adding filters or configurations to your query:

```
QueryState query = getDataSource(request).getQueryState();
query.addFunction(new ResultsConfig());
query.addFunction(new RecordFilter("Region:Midwest"));
QueryResults results = getDataSource(request).execute(query);
```

If you need to make modifications to your query that can't be represented on a `QueryState`, you can use `ENEQuery` instances directly:

```
DataSource ds = getDataSource(request);
ENEQuery eneQuery = ds.createENEQuery();
//modify query...
ENEQueryResults eneResults = ds.execute(eneQuery);
```

When you need to update a data source's state so that all associated components are updated, you cannot use `ENEQuery` instances—you must use `QueryState` instances.

```
DataSource ds = getDataSource(request);
QueryState query = ds.getQueryState();
```

```
query.addOperation(new RecordFilter("Region:Midwest"));
ds.setQueryState(query);
```

Chapter 4

About Discovery Framework logging

The Discovery Framework uses the Apache `log4j` logging utility.

Modifying logging in your Discovery Framework components

There are two ways to modify logging for your Discovery Framework components.

From a developer's perspective, you will most frequently need to adjust a logging verbosity level for a given class or class hierarchy. The easiest way to do this is through the Liferay Control Panel. If you need to modify logging in a more complex manner, or want to change default settings, you can modify the `portal-log4j-ext.xml`.

Both of these methods are described in this section.

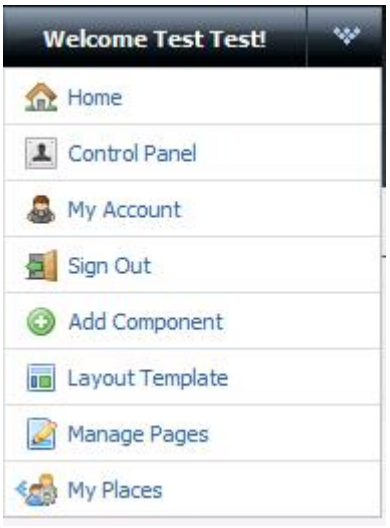
Note: Using either of these methods causes Liferay to adjust the log verbosity for both `log4j` and the Java Utility Logging Implementation (JULI). Code using either of these loggers should respect this configuration.

Adjusting the logging verbosity level in the Control Panel

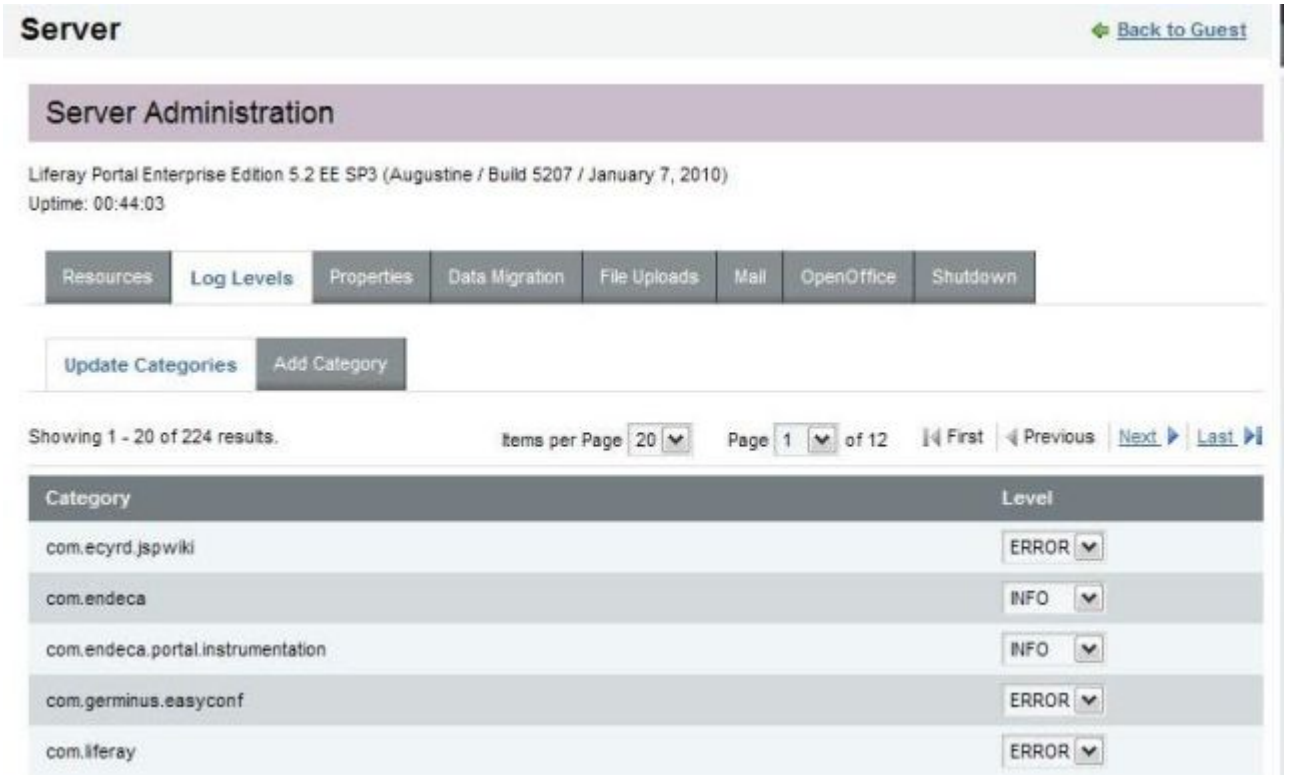
The easiest way to dynamically adjust logging verbosity levels for any class hierarchy is from the Liferay Control Panel.

To adjust logging verbosity in the Liferay Control Panel:

1. In the Discovery Framework, point the cursor at the Dock in the upper-right corner of the page. The Dock is labeled "Welcome <*user name*>!"



2. From the drop-down menu, choose **Control Panel**.
3. From the **Control Panel** tool menu, choose **Server Administration**.
4. In the **Server Administration** pane, choose the **Log Levels** tab.



5. Scroll to find the class hierarchy you want to modify, and then adjust the logging level in the drop-down list. The available options are:
   - OFF
   - FATAL
   - ERROR
   - WARN
   - INFO

- DEBUG
- ALL

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

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

Note: By default, Endeca sets log levels for `com.endeca` and `com.endeca.portal.instrumentation`. You can adjust these levels. In addition, you can set the verbosity for a specific class or package by using the **Add Category** tab.

Modifying portal-log4j-ext.xml

Liferay's primary log configuration is managed in the `portal-log4j.xml` file (which is packed inside the portal application's `WEB-INF/lib/portal-impl.jar`).

As with many other configuration files, Liferay provides administrators with a second configuration file, `portal-log4j-ext.xml` (located in the portal application's `/WEB-INF/classes/META-INF/` directory), which can be used to override settings in the main `portal-log4j.xml` file.

Both of these files are in standard `log4j` XML configuration format. Both files allow creating and modifying appenders, binding appenders to loggers, and adjusting the default log verbosity of different classes and packages. By default, the Endeca override file `portal-log4j-ext.xml` specifics a log verbosity of INFO for the `com.endeca` and `com.endeca.portal.instrumentation` packages. Endeca does not override any of the default log verbosity settings specified for non-Endeca components packaged in `portal-log4j.xml`.

Setting up logging for your Discovery Framework application

The Discovery Framework uses the Apache `log4j` logging utility.

Liferay's primary log configuration is managed in `portal-log4j.xml` file (which is packed inside portal application's `WEB-INF/lib/portal-impl.jar`). As with many other configuration files, Liferay provides administrators with a second configuration file, `portal-log4j-ext.xml`, which can be used to override settings in the main `portal-log4j.xml` file.

Both of these files are in standard log4j XML configuration format. Both files allow creating and modifying appenders, binding appenders to loggers, and adjusting the default log verbosity of different classes/packages. By default, the Endeca override file specifics a log verbosity of INFO for the `com.endeca` and `com.endeca.portal.instrumentation` packages. Endeca does not override any of the default log verbosity settings specified for non-Endeca components packaged in `portal-log4j.xml`.

The Endeca log configuration specifies three appenders. The main root logger prints all messages to two locations: the console, which is typically redirected to the application server's output log (`catalina.out` in Tomcat and `SystemOut.log` in WAS), and a file called `df.log`. That file is specified relative to the working directory. If not adjusted, the `df.log` file can typically be found in one of the following locations:

1. If Tomcat was started by running the `startup.bat` or `startup.sh` script, the log is found wherever the script was run. For example, if you navigate to `tomcat-<version>/bin` and execute the startup script, your logs appear in `tomcat-<version>/bin/df.log`.
2. If Tomcat was registered and started as a Windows service, the log files may be located in `C:\Windows\System32\df.log` or `C:\Windows\SysWOW64\df.log`.
3. If Tomcat is a server inside of Eclipse, the log files may be located in the root of the Eclipse directory (such as `C:\eclipse\df.log`).
4. If running WAS 6.1 or 7, the log files may be located relative to the profile's working directory (such as `/localdisk/WAS/AppServer/profiles/AppSrv01/df.log`).

In addition to the console and `df.log` appenders, Endeca also provide a second file appender for capturing metrics logging. This appender creates a file called `df-metrics.log`, which is generated in the same location as `df.log`. All log entries produced by classes in `com.endeca.portal.instrumentation` are routed to `df-metrics.log`—they are not printed to the console or to `df.log`. By keeping performance information separate, administrators can easily distinguish server logs from performance logs, and can easily run analysis scripts on the performance logs.

Note: For further details on `log4j` logging in Liferay, see the *Liferay Portal Administrator's Guide*.

About log4j.properties files

This topic describes the different versions of the `log4j.properties` file.

The version of the `log4j.properties` file that is located in `common/endorsed/log4j.properties.jar` is used to configure logging for the Tomcat bundle. The file ensures that there is some preliminary `log4j` configuration, because `log4j` is initialized before the Discovery Framework in the Tomcat bundle. This `log4j.properties` file provides minimal configuration, which ensures that initial messages are logged to the console in the same format as the default configuration in `portal-log4j-ext.xml`. The settings in the `log4j.properties` file only affect a small number of messages printed as the server is starting. Once the Discovery Framework starts and loads its XML configuration file, it overrides the settings in the `log4j.properties` file. Therefore, it should not be necessary for administrators to modify this properties file.

In addition, all deployed portlets, as well as the Discovery Framework application itself, have their own `log4j.properties` files, located in `WEB-INF/classes`. Because the Discovery Framework uses XML configuration files, these properties files have no effect.

Learning about log4j

This topic provides links to additional information about `log4j`.

The *Apache log4j site* provides general information about `log4j`, along with documentation.

For more information about `log4j` logging in Liferay, see the *Liferay documentation*, including the *Liferay Portal Administrator's Guide*.

Chapter 5

Getting started with the Discovery Framework

This section describes how to launch and configure the Discovery Framework and begin to work with it.

Starting the Discovery Framework

You start the Discovery Framework by starting your application server, going to the portal in your Web browser, and logging in.

The default login is `test@endeca.com`, and the default password is `test`.

Getting started checklist

Keep the following points in mind when you start up the Discovery Framework:

- Before logging in to the Discovery Framework for the first time, make sure you have a data source in place. For more information, see the topic "Creating a `default.json` file" later in this section.
- Make sure to edit the host and port settings in `default.json` to match your configuration.

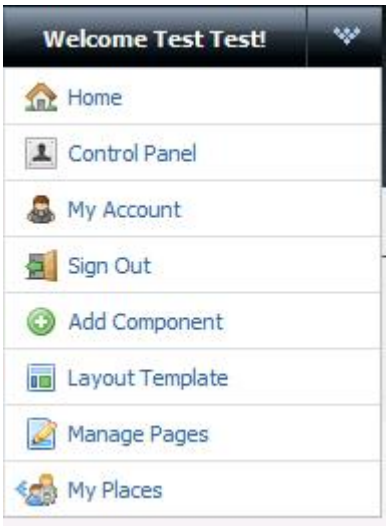
Accessing the Control Panel

After logging in to the Discovery Framework, you may also want to access edit controls. This is done through the **Control Panel**.

The **Control Panel** provides access to a wide range of edit controls, including managing accounts, adding new users, and monitoring performance. For full documentation on **Control Panel** capabilities, see the *Liferay Portal Administrator's Guide*. To access a free PDF download of this guide, go to *http://www.liferay.com* and navigate to Documentation.

To access the **Control Panel**:

1. Point the cursor at the **Dock** in the upper-right corner of the page. The **Dock** is labeled "Welcome *<user name>*!"



2. From the drop-down menu, choose **Control Panel**.

About Framework Settings

Many settings related to Discovery Framework can be adjusted from the **Framework Settings** section of the **Control Panel**.

Configurable settings include the following:

- **df.cordaImageAuthEnabled:** Controls whether Corda chart image URLs are secured.

  Note: The Corda image authorization feature is in Beta for this release and is not supported.

- **df.cordaServerExternalUrl** and **df.cordaServerInternalUrl:** The externally and internally accessible URLs of the Corda Server, which is used by the **Chart** component.
- **df.cordaServerRedirectorUrl:** The URL of the Corda Redirector, which can be used by the **Chart** component.
- **df.dataSourceDirectory:** The directory on disk from which to load data source definition files.
- **df.deepLinkPortletName:** The name of the deep link component.
- **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.maxExportAnalyticsRecords:**The maximum allowable number of Analytics records that can be exported.
- **df.maxExportBaseErrors:** The maximum allowable number of non-Analytics records that can be exported.
- **df.mdexStateManager:** The fully-qualified class name to use for the MDEX State Manager.
- **df.mdexSecurityManager:** The fully-qualified class name to use for the MDEX Security Manager.

The default values of these settings are created automatically upon first use. You cannot add or delete settings from the **Control Panel**—you can only edit them. Settings only appear after the feature(s) that use them have been executed at least once. For example, if you have never used the **Chart** component, the Corda Server URL settings will not appear.

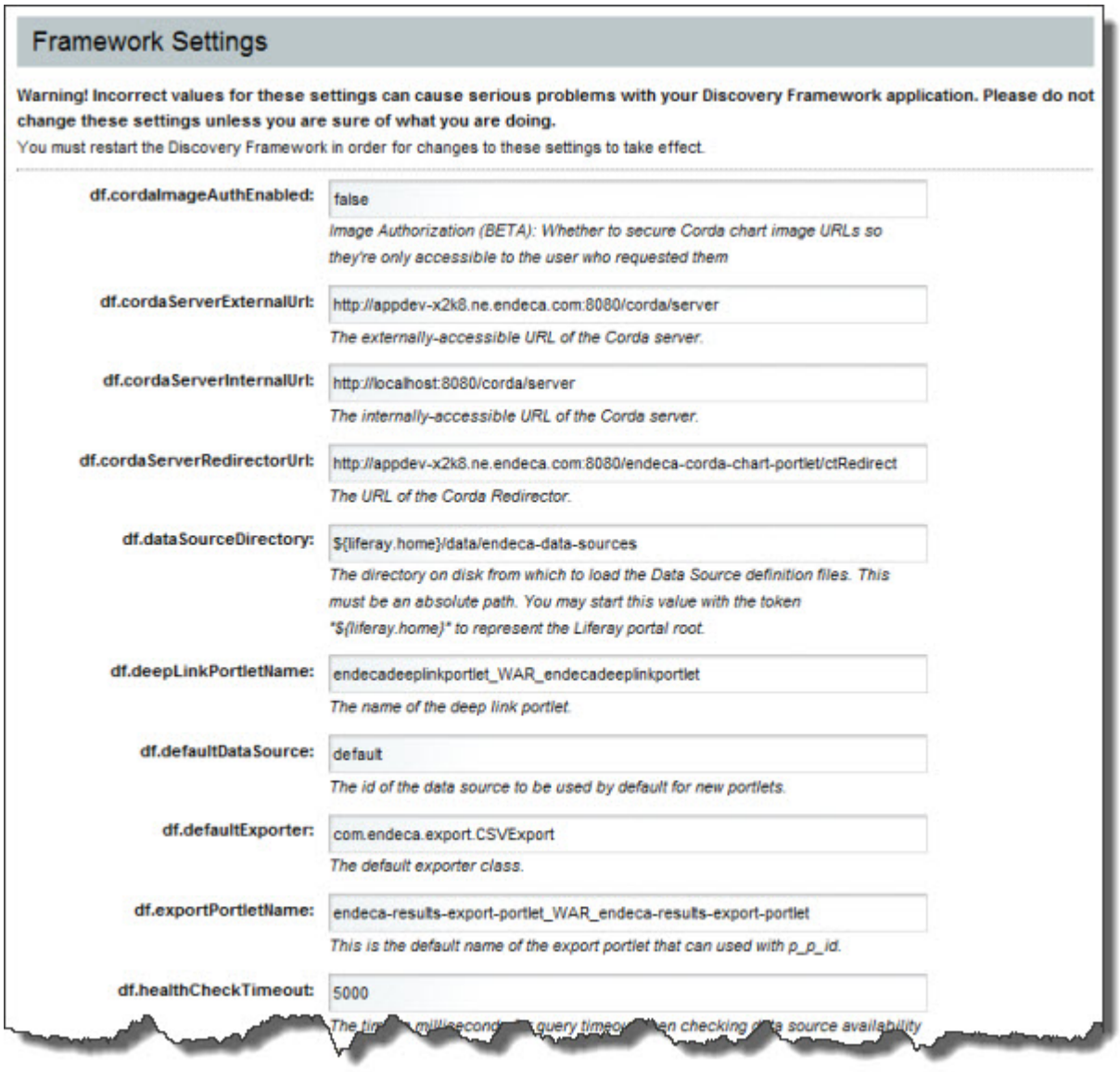
Modifying Framework Settings

You modify **Framework Settings** in the Control Panel.

Important: Take care when modifying these settings, as incorrect values can cause problems with your Discovery Framework application.

To modify **Framework Settings**:

1. In the Discovery Framework, point the cursor at the **Dock** in the upper-right corner of the page. The **Dock** is labeled "Welcome <user name>!"
2. From the drop-down menu, choose **Control Panel**.
3. In the **Discovery Framework** section of the **Control Panel Portal** menu, choose **Framework Settings**.



4. Make your modifications and then click **Update Settings**.

5. Restart the Discovery Framework so your changes can take effect.

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

Adding Endeca standard components

The Discovery Framework contains several Endeca standard components. These components make it possible for you to add Endeca functionality to your application.

To add an Endeca component to your Discovery Framework application:

1. Point the cursor at the **Dock** in the upper-right corner of the page.
2. In the drop-down menu, select **Add Component**.
   The**Add Component** dialog box opens.
3. In the **Add Component** dialog box, expand the Endeca category.
   A list of the available Endeca components appears.
4. Drag the components you want to include, one by one, into the main page layout.

Chapter 6

Other installation tasks

This section discusses some other installation tasks related to your Discovery Framework installation.

Using a different database

The Liferay portal server uses a relational database to store configuration and state, such as portlet preferences, user permissions, system settings, and more.

By default, Liferay uses Hypersonic (HSQL), which is an embedded database running inside the Java virtual machine. HSQL is useful for standing up a Liferay instance very quickly, but must NOT be used in production due to performance issues and its inability to support clustered Liferay instances.

For instructions on switching to another supported database system, see the *Liferay Portal Administrator's Guide*. Keep the following details in mind:

- The Discovery Framework ships with a `portal-ext.properties` file (in the portal distribution's root directory). You can modify this file instead of creating a new one.
- Endeca has tested the Discovery Framework on MySQL and DB2. Other databases are expected to work but have not been explicitly tested.

Overview of switching to a different database

This topic provides a high-level overview of the steps involved in switching from the default Hypersonic database to the production RDBMS of your choice.

Note: Because the details vary from database to database, this topic only provides a high-level overview of this process. For detailed information, see the *Liferay Portal Administrator's Guide*.

To switch to a different database:

1. Install and verify that your database is working.
2. Create a new empty database or schema for the Liferay portal.
3. Create a database user for the Liferay portal.
4. Grant that user access to the appropriate database/schema, with privileges to create tables, alter schemas, and so on in that database. Ensure that the user has remote access from the Liferay application servers.
5. Stop Liferay if it is running.

6. Edit the `portal-ext.properties` file. In the JDBC section, comment out the settings for Hypersonic, and uncomment the settings for your database.
7. Edit the settings for your database of choice, adding the appropriate username and password and editing the JDBC connection string as necessary.
8. Start the Discovery Framework. Monitor its logs to ensure for any error messages while connecting to the database and creating tables.
9. After tables have been created and you have validated Liferay is running, you may remove the liferay user's alter table privileges. Note you may have to add these back later if you upgrade Liferay or install components that require schema changes.

Installing Corda

The **Chart** component requires the installation of Corda charting software. Endeca recommends deploying Corda Server as a servlet when using it with the Discovery Framework. If you plan to use the **Chart** component, make sure to download `corda-<version>.zip` and deploy the Corda Server servlet in that package.

Note: The Corda image authorization feature is in Beta for this release and is not supported.

Obtaining the Corda software

You download the Corda software package, along with the rest of the Discovery Framework, from the EDeN downloads page.

The Corda software is packaged in the `corda-<version>.zip` file.

About the Corda Server servlet

This topic describes the Corda Server servlet shipped with the Discovery Framework.

The Corda Server servlet is a Java servlet version of the Corda Server. It is designed to run on, and be accessed through, Java-enabled application servers such as Tomcat and WAS. Because it is packaged as a servlet, you do not need to run this version of the Corda Server as a separate process over a separate server port.

For details on how the Corda Server servlet is packaged, see the *Corda documentation*.

Note: If you choose to deploy Corda as a server, rather than the recommended approach of deploying Corda as a servlet in an existing application server, see the Corda installation instructions in the *Endeca Platform Services Installation Guide*.

Important: Deploying the Corda servlet on the same Tomcat server as the Discovery Framework is intended for development purposes only. You should install Corda on a separate application server (or as a standalone server) for production use. If you purchased the Corda charting module (also known as the Advanced Visualization for Java and .NET module), your license entitles you to run a single production instance of the Corda server (whether deployed as a servlet or as a standalone server).

About the Discovery Framework implementation of Corda

All communication between the Discovery Framework and Corda occurs through the Corda Redirector. The location of the Corda Server servlet (or Corda Server) is configurable, allowing it to be deployed locally for development and remotely for production.

The Corda Redirector module is a proxy that allows the browser requests that embed generated chart images to be routed through the application server (in this case Liferay), rather than going directly to the Corda server. All requests for Corda images are made to the Corda Redirector module.

This new redirector-based deployment model ensures a greater level of control over network infrastructure. It enables system administrators to control network access, by allowing direct access *only* to the application server(s).

For more information about the Corda Redirector, see the *Corda documentation*.

Deploying the Corda Server servlet in an application server

The Corda Server servlet can be deployed by following the standard servlet deployment procedure for the application server in question.

Before deploying Corda, you must configure it. The following topics describe this process.

Note: Deploying the `corda.war` requires the use of an archiving tool to expand the archive file. This Java archive can be expanded and re-packaged with Java's jar tool or with a zip utility.

Extracting the corda-<version>.zip file

After you download the `corda-<version>.zip` file, you need to extract it to a temporary location in order to access the `corda.war` file.

To prepare the `corda.war` file for modifications:

1. Unzip the `corda-<version>.zip` file to your hard drive to make the `corda.war` file available.
2. Expand the `corda.war` file.
   This Java archive can be expanded and re-packaged with Java's jar tool or with a zip utility.

Configuring Corda to allow connections from other hosts

The Corda Server servlet is set up by default to accept incoming requests from any host. You must configure it in order to allow only incoming requests from the application server that hosts the Discovery Framework.

Note: In the instructions below, the paths are relative to the location to which the `.war` was extracted.

To change the Corda configuration:

Update the `WEB-INF/classes/Corda60/config/path.xml` configuration file in `corda-web.war` to include entries for the hosts that need to embed charts powered by the Corda Server.

For example, entries like the following may be added to enable access from a specific host, from a range of domain names, or from a range of IP addresses, respectively:

```
<PathMaps Version="1.0">
   <Map Name="DefaultRead" Path="./*" Action="Load"/>
   <Map Name="DefaultSave" Path="./images/*" Action="Save"/>
   <Map Name="ValidDomain" Path="127.0.0.1" Action="allowDomain"/>
   <Map Name="ValidDomain" Path="localhost" Action="allowDomain"/>
   <Map Name="ValidDomain" Path="*" Action="allowDomain"/>
</PathMaps>
```

The last line, highlighted in bold, leaves Corda open to requests from anywhere. For this reason, you should replace **"*"** with the IP address (or range of addresses) allowed to access it (such as, for example, “192.168.*”). Specifically, you should add the location of the application server hosting the Discovery Framework.

Refer to *Corda’s documentation* for details about this configuration file.

Adding or removing PCXML templates

PCXML templates are XML-based templates that describe and define the charts and maps used by Corda. This topic describes how you can deploy the PCXML templates distributed with the Corda Server servlet.

Note: In the instructions below, the paths are relative to the location to which the WAR was extracted.

To deploy the PCXML templates, update the servlet with the new PCXML templates. The steps to update the servlet may differ, depending on the application server and configuration used when deploying the servlet. In all cases, this can be accomplished by updating `corda.war` with the required changes and repeating the steps in the topic "Deploying the corda.war file" to deploy the modified `.war` file.

The PCXML chart templates are located in the following location in `corda.war`: `WEB-INF/classes/Corda60/chart_root/apfiles`.

Note: Adding new PCXML chart templates requires updates to the `Chart` component to use the newly deployed PCXML files. For more information, see the topic "Configuring Chart to use PCXML templates."

Deploying the corda.war file

After configuring Corda, you must repackage the `.war` file and then deploy it.

Note: Hot-deploying the Corda Server servlet into Liferay's deploy directory is not supported.

Deploying on a Tomcat server:

1. Re-zip the modified `corda.war` file.
2. Deploy the file into Tomcat's `webapps` directory (such as `/path/to/tomcat-<version>/webapps`).
3. Restart Tomcat.

Depending on your Tomcat configuration, the servlet container may unpack the `.war` archive, or it may operate directly from the archive. If you plan to modify Corda configuration files further, or to

deploy or modify PCXML chart templates, you may prefer to unpack the `corda.war` archive, to provide easier access to files inside the archive.

Deploying on WebSphere Application Server:

1. Re-zip the modified `corda.war` file.
2. Use the IBM Integrated Solutions Console, Deployment Manager, or `wsadmin` utility to deploy `corda.war`. The servlet should be deployed as an enterprise application with context root `/corda`.

Confirming the Corda Server servlet deployment

After deploying the Corda Server servlet, you should ensure that it is running.

The server log should display messages similar to the following example when the Corda Server servlet starts successfully:

```
Corda Server (PopChart) Version 6.0.735
  PopChart: Valid Key, OEM build for: ENDECA.
  OptiMap: No key entered, or key invalid.
  Highwire: No key entered, or key invalid.
  Cluster: No key entered, or key invalid.

Copyright 1997 - 2006, Corda Technologies, Inc. (www.corda.com) Protected
by U.S
. Patent 5,933,830. Other patents pending.

server_root: /Corda60
chart_root: chart_root
Cache Segment Size: 0
Password is Enabled, Required for Save
Maximum Threads: 64
Default Image Type is: Flash
Auto Detect PNG Support. Compression Mode: DEFAULT
```

In addition, you can test the Corda Server servlet in a browser, as follows:

1. Access the following URL, using the name of your server rather than `server.example.com`:

   ```
   http://server.example.com:8080/corda/server
   ```

   A empty white box with a gray background displays.
2. Confirm that the title bar says **Corda Server (PopChart) Servlet Version 6.0.735**.
3. Access the following URL, using the name of your server rather than `server.example.com`:

   ```
   http://server.example.com:8080/endeca-corda-chart-portlet/ctRedirect
   ```

   The same empty white box and title bar mentioned above should appear.

Updating the Chart component with changes to the Corda Server servlet

By default, the `Chart` component is configured to look for the local instance of the Corda Server servlet, and fails if the servlet is not deployed.

You can change where the `Chart` component looks for Corda, if you are installing the servlet on a non-localhost machine or you have a Corda server already running elsewhere.

By default, the `Chart` component is configured to use a Corda Server deployed as a servlet on the same application server as the Discovery Framework. This is a convenient configuration for single-server deployments and development and demonstration environments. However, production environments

(especially those with clustered application servers) may require alternate configuration to specify a separate location for the Corda Server.

The internal and external hosts used by Corda Server differ in cases where Corda is deployed as a standalone server and cases where Corda is deployed as a servlet.

- When deployed as a servlet, the internal and external URL typically take a form similar to `http://server.example.com:[app server port]/corda/server`.
- When deployed as a standalone server, the internal and external hosts typically take a form similar to `http://cordaserver.example.com:[corda server port]/` (where the Corda server port differs for the internal and external URLs). For details, refer to the *Corda documentation*.

Configuring the location of the Corda Server

You can configure Corda Server servlet location in the Discovery Framework's **Control Panel**.

Important: Because the **Control Panel** only shows settings that exist, in order to be able to edit the Corda Server URL properties, you must put the `Chart` component on a page first.

To configure the Corda Server URL:

1. Point the cursor at the **Dock** in the upper-right corner of the page.
2. In the drop-down menu, choose **Control Panel**.
3. In the **Portal** section of the **Control Panel** navigation panel, select **Discovery Framework Settings**.
4. Update the internal, external, and redirector URLs—`df.cordaServerInternalUrl`, `df.cordaServerExternalUrl`, and `df.cordaServerRedirectorUrl:`—to point to the location (host and port) of the Corda Server servlet. For example:
   ```
   df.cordaServerInternalUrl = http://server.example.com:8080/corda/server
   ```
5. Click **Update Settings**.
6. Restart the Discovery Framework.

Configuring Chart to use PCXML templates

During deployment, you added PCXML templates to your Corda implementation. Now you can configure the **Chart** component to use these templates.

The `endeca-corda-chart-portlet-<version>.war` archive file is included in `components-<version>.zip`.

To configure the **Chart** component's PCXML templates:

Update the `WEB-INF/analytics-portlet-config.xml` file in `endeca-corda-chart-portlet-<version>.war` and add or remove the `CordaChartConfiguration` elements.

Note: This Java archive file can be expanded and re-packaged with Java's jar tool or with a zip utility.

For example, to add a new chart called 3D Pie with PCXML template `3DPie.pcxml`, update the file to include the following XML:

```
<bean id="3dPieChart" class="com.endeca.portlet.corda.CordaChartConfigu¬
ration">
  <property name="chartDisplayName" value="3D Pie" />
```

```
  <property name="pcxml" value="3DPie.pcxml" />
</bean>
```

This configuration causes the `Chart` component to display an additional option on its **Preferences** panel, allowing the use of 3D Pie as a chart style.

Troubleshooting Corda

The following section provides troubleshooting information for Corda.

Change in Corda behavior in Discovery Framework

Starting in Discovery Framework version 1.3.1, the Corda splash screen no longer renders upon entering the base URL.

In addition, URLs such as @_FILE URL are now restricted, to prevent cross-site scripting vulnerabilities.

Problems rendering a chart

When attempting to render a chart, the `Chart` component may fail to reach the Corda Server at the specified host and port.

If the host and port configuration is correct, you may need to configure Corda to allow connections from the application server hosts that are hosting the Discovery Framework. For details, see the topic "Configuring Corda to allow connections from other hosts," which appears earlier in this section.

Known issue with JVM crashes related to Corda Server servlet and JIT compilation

There is a known issue with JVM crashes that is related to the Corda Server servlet and JIT compilation on the latest version of Sun's JVM (specifically, version 1.6_21).

When this JVM crash occurs, the JVM error log typically notes that the active thread is a compiler thread, similar to the following:

```
Current thread (0x00002aab04077800):  JavaThread "CompilerThread1" daemon
 [_thread_in_native, id=2353, stack(0x0000000041123000,0x0000000041224000)]
```

The compilation task being performed is related to the method `com.corda.b.dc.a()`, similar to the following:

```
Current CompileTask:
C2:4112      com.corda.b.dc.a(B)Z (559 bytes)
```

This issue can be resolved by disabling JIT compilation with the following JVM argument:

```
-XX:CompileCommand=exclude,com/corda/a/dc,a -XX:CompileCommand=ex¬
clude,com/corda/b/dc,a
```

There is an expected, albeit minimal, performance degradation associated with disabling JIT compilation. You should be aware of the change and may want to perform additional testing to ensure adequate performance.

Note: This issue has not been reproduced on the latest version of Sun's 1.5 JVM.

Note: This issue is not know to exist on IBM's JVM and has not been reproduced on it.

Uninstalling the Discovery Framework

To uninstall the Discovery Framework, remove the packages and directories that you installed.

If your previous deployment was installed in an existing application server, follow the appropriate steps to stop and uninstall the earlier DF application from the application server before installing DF 1.3. For important details on Discovery Framework migration, see the topic "Upgrading from a previous version of the Discovery Framework," located in the section "Before you install."

Note: If you are installing a component-only release, you do not need to uninstall first. For details, see the topic "Installing the component-only package," located in the section "Installing the Discovery Framework."

Note: You cannot have multiple versions of the Discovery Framework installed on the same machine simultaneously.

Index

6.1.4 IAP 39
6.1.4 IAP, Tomcat 39

A

accessing
    the Control Panel 63
adding
    data sources 48
    Endeca components 66
    PCXML templates 70
aggregated records
    configuring in a data source 48

C

Chart component
    changing Corda Server source 71
component install prerequisites 40
component-only package
    installing 39
    installing on Tomcat 41
    installing on WAS 41
components
    changing data source for 48
components, in this version 10
configuring
    the Corda server location 72
    Tomcat 5.5 21
confirming
    Corda Server servlet deployment 71
context root
    changing for the Linux Tomcat bundle 19
    changing for the Windows Tomcat bundle 17
Control Panel
    accessing 63
    adjusting logging level in 59
Corda
    adding PCXML templates 70
    configuring to allow connections from other hosts 69
    confirming the servlet deployment 71
    deploying the servlet 69
    Discovery Framework implementation of 69
    image authorization Beta 68
    installing 68
    obtaining 68
    using with SSL 69
Corda server
    configuring location 72
Corda, deploying corda.war 70
Corda, extracting the zip file 69
Corda, fixing connection problems
    Corda
        troubleshooting 69, 73
    troubleshooting
        Corda 69, 73
Corda, splash screen rendering changes 73
creating
    Liferay Home directory for WAS 6.1 27
    Liferay home directory on WAS 7 34
    new QueryFunction classes 55

D

data source
    default 47
data source configuration
    about 45
    deploying on WAS 6.1 29
    deploying on WAS 7 36
data source example 45
data source limitation on MDEX 7 47, 51
data source relationships 51
data source results
    obtaining 57
data source role-based security 50
data sources
    about 45
    adding 48
    changing for a component 48
    configuring aggregated records in 48
    connecting to a secured MDEX Engine 49
    samples 47
    troubleshooting 48
databases
    switching 67
default data source 47
deploying
    components in WAS 6.1 29
    components in WAS 7 37
    data source configuration on WAS 6.1 29
    data source configuration on WAS 7 36
deploying Discovery Framework dependency libraries on WAS 6.1 24
deploying Discovery Framework dependency libraries on WAS 7 32
deploying the Corda Server servlet 69
deploying the standalone portal WAR on WAS 6.1 26
deploying the standalone portal WAR on WAS 7 33
Discovery Framework
    about settings for 64
    deploying and starting on Tomcat 5.5 22
    downloading 16, 40, 42
    interaction with Liferay Portal 9

Discovery Framework *(continued)*
license key 19, 23, 31, 38
Linux installation steps 18
modifying logging 59
modifying settings 65
obtaining more information 13
QueryFunction classes 51
starting 63
uninstalling 74
Windows installation steps 16
Discovery Framework documentation 42
downloading Corda software 68

E

Eclipse
adding new .jars to your build bath 57
edit controls
turning on 63
Endeca components
adding 66
entering the license key 19, 23, 31, 38
examples
data source 45

F

features in this version 10

H

HTTPS
connecting to an MDEX Engine 49

I

IAP 6.1.4
using the Discovery Framework with 39
IAP 6.1.4, WAS 39
inheritSecurity filter 50
installing
component-only package 39
Discovery Framework on Linux 18
Discovery Framework on Tomcat 5.5 19
Discovery Framework on WAS 7 31
Discovery Framework on WebSphere Application Server 6.1 23
Discovery Framework on Windows 16
full Discovery Framework package 15

J

JSON
about data source syntax in 45
connecting to MDEX 7 46
JVM crashes
in Corda 73

L

Liferay home directory
creating for WAS 7 34
Liferay Home directory
creating for WAS 6.1 27
Liferay portal
accessing documentation for 13
Liferay Portal
interaction with 9
Linux Tomcat bundle
changing the context root 19
log4j
obtaining more information 62
log4j.properties files 62
logging
setting up 61
ways to modify 59
logging verbosity
adjusting 59

M

manually deploying generated .war files in WAS 6.1 30
manually deploying generated .war files in WAS 7 37, 41
MDEX 6.1.4
using the Discovery Framework with 39
MDEX 7
data source limitation on 47, 51
setting JSON file for data sources 46
MDEX 7 support in this version 12
MDEX Engine
sample data sources 47
migrating from an earlier version 13
modifying
PCXML templates 70
portal-log4j-ext.xml 61

O

obtaining additional information 13
obtaining data source results 57
obtaining the Discovery Framework software 16, 40
overview
of deploying on WAS 6.1 24
of deploying on WAS 7 32
of switching databases 67
Tomcat 5.5 deployment 20
overview of package 9

P

package overview 9
parent and child data sources 51
parentDataSource filter 50
PCXML
adding templates 72
removing templates 72

PCXML templates
    adding 70
portal-ext.properties
    configuring for WAS 6.1 27
    configuring for WAS 7 35
    editing for WAS 6.1 27
    editing for WAS 7 34
    updating the WAS 6.1 application with 28
    updating the WAS 7 application with 35
    uploading to WAS 6.1 28
    uploading to WAS 7 36
portal-ext.properties, example for WAS 6.1 28
portal-ext.properties, example for WAS 7 36
portal-log4j-ext.xml
    about 61
prerequisites 11, 12

Q

query function classes, implementing 56
QueryFunction classes
    adding new .jar to the Eclipse build path 57
    creating your own 55
    in the Discovery Framework 51
QueryFunction classes, using custom 56
QueryFunction, implementing new 55

R

relational database, changing
    database
        changing 67
    Hypersonic
        replacing with another database 67
rolePermissions filter 50

S

sample data sources 47
secured MDEX Engine
    connecting to 49
securityEnabled filter 50
securityFilters filter 50
setting up log4j 61
settings
    about 64
    modifying 65
standalone portal WAR
    deploying on WAS 6.1 26
    deploying on WAS 7 33
starting the Discovery Framework 63
system requirements 11, 12

T

Tomcat
    installing component-only upgrades on 41
Tomcat 5.5
    deploying and starting on 22
    deploying the dependency libraries 20
    high-level deployment overview 20
    installing 20
    installing Discovery Framework on 19
    modifying configuration 21
troubleshooting
    WAS 6.1 deployment 31
    WAS 7 deployment 39
troubleshooting data sources 48
turning on edit controls 63

U

uninstalling
    the Discovery Framework 74
updates to the Discovery Framework 39
upgrading the Discovery Framework 13

V

verbosity
    logging 59

W

WAS
    deploying dependency libraries on WAS 6.1 24
WAS 6.1
    configuring portal-ext.properties for 27
    deploying components in 29
    editing portal-ext.properties 27
    extracting the standalone portal WAR 25
    high-level deployment overview 24
    manually deploying generated .war files 30
    manually uploading portal-ext.properties to 28
    starting the application 29
    troubleshooting 31
WAS 7
    configuring portal-ext.properties for 35
    deploying components in 37
    deploying dependency libraries on WAS 7 32
    editing portal-ext.properties 34
    extracting the standalone portal WAR 33
    high-level deployment overview 32
    installing 31
    manually deploying generated .war files 37, 41
    manually uploading portal-ext.properties to 36
    starting the application 36
    troubleshooting 39
WAS 7, deploying components 30, 37
WAS 7, Liferay component pre-processing 29, 37
WebSphere
    installing component-only upgrades on 41
WebSphere Application Server 6.1
    installing Discovery Framework on 23
Windows Tomcat bundle
    changing the context root 17

wsadmin
using to deploy on WAS 6.1 30
wsadmin *(continued)*
using to deploy on WAS 7 38, 42