



Cúram 8.1.3

Deploying on Oracle WebLogic Server Guide

Note

Before using this information and the product it supports, read the information in [Notices on page 41](#)

Edition

This edition applies to Cúram 8.1, 8.1.1, 8.1.2, and 8.1.3.

© Merative US L.P. 2012, 2024

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.

Contents

| | |
|---|------------|
| Note | iii |
| Edition | v |
| 1 Deploying on Oracle WebLogic Server | 9 |
| 2 Building EAR files | 11 |
| 2.1 The Enterprise Application..... | 11 |
| Building the Application.ear File..... | 11 |
| Under the Hood..... | 11 |
| Contents of the <i>application.ear</i> file..... | 12 |
| 2.2 The Web Services Application..... | 13 |
| Building the Web Services.ear File..... | 13 |
| Under the Hood..... | 14 |
| Contents of Web Services.ear File..... | 14 |
| Web Service WSDL..... | 15 |
| 2.3 Multiple EAR files..... | 15 |
| 3 Application server configuration | 17 |
| 3.1 WebLogic Server configuration..... | 17 |
| Configuring a web server plug-in in Oracle WebLogic Server..... | 18 |
| 3.2 Security configuration for Oracle WebLogic Server..... | 19 |
| 3.3 Time Zone Configuration..... | 20 |
| 3.4 Starting and Stopping WebLogic Servers..... | 20 |
| Start a WebLogic Server..... | 20 |
| Stop a WebLogic Server..... | 21 |
| Restart a WebLogic Server..... | 21 |
| Server Logging..... | 22 |
| 4 Deployment | 23 |
| 4.1 Deployment..... | 23 |
| Deploy an Application..... | 23 |
| Change SYSTEM Username..... | 24 |
| Undeploy an Application..... | 24 |
| 4.2 Pre-compiling JSPs..... | 24 |
| 4.3 Testing Deployment..... | 25 |
| 5 Manual WebLogic Server configuration | 27 |
| 5.1 Creating a WebLogic Server Domain..... | 27 |
| Domain Configuration..... | 27 |
| 5.2 Java Virtual Machine Configuration..... | 28 |
| 5.3 Set up Database Driver..... | 29 |
| 5.4 Starting the WebLogic Administration Service..... | 29 |

| | |
|--|-----------|
| 5.5 Database Configuration..... | 29 |
| 5.6 Setup Security..... | 31 |
| 5.7 Setup Transport Layer Security..... | 33 |
| 5.8 Setup JMS..... | 34 |
| 5.9 Queue Setup..... | 34 |
| 5.10 Queue Connection Factory Setup..... | 36 |
| 5.11 Topic Setup..... | 37 |
| 5.12 Topic Connection Factory Setup..... | 38 |
| 5.13 Manual Deployment..... | 38 |
| Notices..... | 41 |
| Privacy policy..... | 42 |
| Trademarks..... | 42 |

1 Deploying on Oracle WebLogic Server

To deploy Cúram on Oracle WebLogic Server, you must configure the server and deploy the application EAR files.

It is assumed that Oracle WebLogic Server is installed, see .

2 Building EAR files

The main step before deployment of an Cúram application is to package it into enterprise archive (.ear) files. The application (web client, server) and web services application are all packaged into separate .ear files.

The Server Development Environment for Java™ (SDEJ) provides Apache Ant targets for packaging .ear files.

Before running build targets in the following section, you must set WLS_HOME environment variable, and the previously set environment variables, see .

The WLS_HOME variable should point to the *server* directory of the *Oracle WebLogic Server* installation, for example: `Windows d:\weblogic\server` or `Linux /opt/weblogic/server`.

2.1 The Enterprise Application

The following sections describe the process for building the application .ear (Enterprise ARchive) file. They also provide information on what happens "under the hood" and on the contents of the EAR file.

Building the Application.ear File

The following target should be executed from the root directory of the project to create the .ear file for WebLogic Server:

```
build weblogicEAR
```

This target will create a ready to install .ear file, `<SERVER_MODEL_NAME>.ear`, located in `<SERVER_DIR>/build/ear/WLS`.

Note: SERVER_MODEL_NAME and SERVER_DIR are environment variables which specify the name of the model in the project and the root directory of the project respectively.

Before executing this target, a fully built Cúram application must be available. For details on how to build a Cúram application, please refer to the *Cúram Server Developer's Guide*.

Note: The EAR file cannot be built for H2 database. For more information on H2 database consult the *Cúram Development Environment Installation Guide*.

Under the Hood

The `weblogicEAR` target takes a number of previously generated Java® files and deployment descriptors and packages them up into an EAR file.

The Java files and deployment descriptors are generated during the build process based on the existence of Business Process Object (BPO) classes, i.e., the methods of *Facade* classes or *WebService* classes and can be called by remote clients.

By default all remote calls to the server are handled by the session bean `curam.util.invoke.EJBMethod`, rather than a session bean per publicly available interface. This bean provides support for application features such as authorization, auditing and tracing. If required it is also possible to generate a Facade interface

Note: The optional build parameter-`Denablefacade=true` turns on the generation of facade code.

Contents of the *application.ear* file

The *.ear* file that is produced has the following structure and contents:

- **META-INF Directory**

The *META-INF* Directory includes the following:

- *application.xml*

This file is automatically generated and lists the mapping of EJB modules to JAR files that are contained in the application.

- *MANIFEST.MF*

This file details the contents of the *.ear* file.

- **Core *.jar* Files**

The version numbers are not listed for the *.jar* files detailed.

The core *.jar* files include:

- *appinf.jar*
- *appinf_internal.jar*
- *coreinf.jar*
- *rules.jar*
- *jde_commons.jar*
- Apache Log4j 2 jar files: *log4j-api.jar*, *log4j2.jar*, and *log4j2-config.jar*
- *commons-pool.jar*
- *commons-codec.jar*
- *commons-discovery.jar*
- *jdom.jar*
- *axis.jar*
- *jaxrpc.jar*
- *saaj.jar*
- *java_cup.zip*

- *InfrastructureModule.jar*
- *InvalidationModule.jar*
- *DBtoJMS.war*
- *ClientModule.war*
- **Facade .jar Files**

These are only present if facade generation has been enabled. All facades defined in the application are packaged into one *.jar* file, *FacadeModule.jar*. This *.jar* file contains the bean implementation classes for the EJB modules that represent the facades. The *.jar* file contains the following files in the *META-INF* directory:

- *ejb-jar.xml*

This file is automatically generated and contains the definition of every EJB module contained in the *.jar* file. All the publicly available methods are listed and the details of the resources available to the EJB modules.

- *weblogic-*ejb-jar.xml**

Generated WebLogic Server specific deployment descriptor.

- *Manifest.mf*

The manifest file, detailing the classpath for the EJB.

- **Other .jar Files**

The other *.jar* files contain the generated and hand crafted code from the application.

These include *struct.jar*, *application.jar*, *workflow.jar*, *implementation.jar* and *properties.jar*. The *properties.jar* file contains the *Bootstrap.properties* file. This is the file containing the machine specific configuration properties for initially getting a connection to the database.

2.2 The Web Services Application

Support is available for the automatic generation of WSDL (Web Service Definition Language) defined Web Services. Application developers can thus combine the power of the Cúram model with the accessibility of web services to produce truly reusable software components.

Building the Web Services.ear File

The following target should be executed from the root directory of the project to create the *.ear* file for web services:

```
build weblogicWebservices -Dprp.webipaddress="address" -
Dprp.contextproviderurl="url" -Dprp.contextfactoryname="factory"
```

Where:

- *address* is the IP address on which the server hosting the web services is listening. The default is *http://localhost:7001*;

- `url` is the URL of the JNDI context provider. This is the address of the server that hosts the Cúram components being made accessible through web services. The default is `t3://localhost:7001;`

Note: The default value for the JNDI context provider uses a WebLogic Server -specific protocol, `t3`, for accessing the JNDI tree.

- `factory` is the JNDI context factory name. The default for this is `weblogic.jndi.WLInitialContextFactory` and should rarely need to be changed.

This target will create a ready to install `.ear` file,

`<SERVER_MODEL_NAME>WebServices.ear` located in `<SERVER_DIR>/build/ear/WLS`.

Note: Before executing this target, a fully built Cúram application, ready for deployment, must exist.

Under the Hood

The `weblogicWebServices` target takes a number of previously generated Java files and deployment descriptors and packages them up into an `.ear` file.

The Java files and deployment descriptors are generated during the build process (see the *Cúram Server Developer's Guide*) based on the web service stereotypes that have been defined in the model. BPO classes should be mapped to server components with a stereotype of `webservice` for this generation to occur. Any server component with a stereotype of `webservice` will be treated as if it also had a stereotype of `ejb`. This is because web service interfaces are wrappers on publicly available BPOs.

Consult the *Cúram Server Modelling Guide* for details on assigning BPOs to server components.

Contents of Web Services.ear File

The web services `.ear` file that is produced has the following structure and contents:

- `META-INF` Directory

- `application.xml`

This file details the core module for the web services application, which is the `webservices.war` file.

- `MANIFEST.MF`

The manifest file which details the contents of the `.ear` file.

- **Web Service .war File**

This file contains support `.jar` files in the `WEB-INF/lib` directory, including:

- `coreinf.jar`

This `.jar` file contains the conversion methods which are used to support the serialization of the complex types used in the interface.

- *axis.jar*

This *.jar* file contains the Apache Axis web services engine.

- *appwebservices.jar*

This *.jar* file contains the wrapper classes which enable the Axis web services to connect to Cúram session bean(s) and the classes for the complex types which are used in the interface to the web services.

- *server-config.wsdd*

The *.wsdd* file is located in the *WEB-INF* directory and contains the web service engine configuration which maps Cúram BPO s to web services.

Web Service WSDL

A Cúram Axis web service exposes its own WSDL once it is deployed.

For instance, if there is a service at the URL:

```
http://localhost:7001/CuramWS/services/MyTestService
```

the WSDL description will be at the URL:

```
http://localhost:7001/CuramWS/services/MyTestService?wsdl
```

The URL

```
http://localhost:7001/CuramWS/services
```

will return a web page that lists all Axis web services deployed and a link to their WSDL files.

The general URL format of the locations above is

```
http://<web-server>:<port-number>/<ServerModelName>WS/services/  
<BPO-name>.
```

2.3 Multiple EAR files

Building an application EAR also takes an optional file to allow for splitting the client components into different WAR and EAR files and also to allow for some more control of some of the EAR configuration and included modules. This file is named *deployment_packaging.xml* and should be placed in your *SERVER_DIR/project/config* directory.

The format of the *deployment_packaging.xml* file is as follows:

```
<deployment-config>
  <ear name="Curam"
    requireServer="true">
    <components>custom,sample,SamplePublicAccess,core</
components>
    <context-root>/Curam</context-root>
  </ear>
  <ear name="CuramExternal">
    <components>SamplePublicAccessExternal</components>
```

```

    <context-root>/CuramExternal</context-root>
    <custom-web-xml>${client.dir}/custom_web_xml</custom-web-xml>
  </ear>
</deployment-config>

```

Figure 1: *deployment_packaging.xml* Sample

Each file can have multiple `<ear>` elements and results in an EAR file being produced in the `SERVER_DIR/build/ear/WLS` directory. The options for each element are:

- name

This option controls the name of the EAR created from the process.

- requireServer

This optional attribute controls whether the server module is included in the EAR file. Valid entries are `true` or `false`. The default value is `false`. If deploying multiple EAR files to one application server, this attribute must be set to `true` for only one EAR file as only one Cúram server module should be deployed per cluster. If `requireServer` is set to `true` for multiple EAR files, then the other EAR files must be deployed in another cluster to avoid conflicts.

- components

This option controls which of the client components get placed into the EAR file. It also controls the component order for the rebuild of the client that will need to take place. Usually the core directory doesn't form part of the component order but on this occasion it is important to add this to qualify whether it should be included in a particular WAR file. Entries here should follow the typical order of components defined in the *Cúram Server Developer's Guide* and should be comma separated.

- context-root

This option forms the Context Root of the WAR module in the *application.xml* deployment descriptor. Entries here should begin with a forward-slash.

- custom-web-xml

This optional element controls whether a custom *web.xml* file should overwrite the standard version in the WAR file. Entries here should be an Apache Ant path to the directory containing the *web.xml* file.

It is possible to use references to environment variables as part of this path. For example, `${client.dir}` can be used to point to the web client directory and `${SERVER_DIR}` can be used to point to the server directory.

For each web client (i.e., WAR file) a separate web client component is required to contain its customizations. In the case of multiple web clients, your `CLIENT_COMPONENT_ORDER` environment variable will include all your custom components; but, separate `<ear>` elements will be required, one for each custom web component (and other components as needed).

As with the standard target, a fully built application must be available. For details on how to build an application, please refer to the *Cúram Server Developer's Guide*.

3 Application server configuration

The configuration of Oracle WebLogic Server is the same on all platforms and the Cúram Server Development Environment for Java™ (SDEJ) provides a number of Ant targets to aid the configuration and management of the installation.

For information about the manual steps done by the configuration scripts, see [5 Manual WebLogic Server configuration on page 27](#).

The configuration target provided by the SDEJ is a simple default configuration and may not be suitable for a production environment.

Note: The `configure` target will overwrite any existing Oracle WebLogic Server domain that it is targeted to configure.

3.1 WebLogic Server configuration

To configure the WebLogic Server, you must set up a data source, a domain, a server, and configure the JMS and security settings. You can do all of these tasks by executing the `configure` target that is provided by the Cúram SDEJ.

The command `build configure` must be executed from the `<SERVER_DIR>` directory to invoke automatic configuration. This target requires that the files `AppServer.properties` and `Bootstrap.properties` exist in the `<SERVER_DIR>/project/properties`.

Note: You can overwrite this default location for the properties file by specifying `-Dprop.file.location` when executing the `configure` target.

directory. For more information about how to set up a `Bootstrap.properties`, see the [3.1 WebLogic Server configuration on page 17](#) details the required contents of the `AppServer.properties` file.

```
## APPLICATION SERVER PROPERTIES

## IT IS VERY IMPORTANT TO USE '/' FOR DIRECTORY PATHS. ##

# Property to indicate WebLogic Server is installed.
as.vendor=BEA

# The username and password for admin server.
# The password must be encrypted.
security.username=<e.g. weblogic>
security.password=<e.g. encrypted password>

# The name of the WebLogic Server Domain Name.
node.name=MYNODE

# The name of the server on which the application will be hosted.
curam.server.name=CuramServer
```

```
curam.server.port=7001

#####
## THE FOLLOWING PROPERTIES ARE FOR WEBLOGIC ONLY ##
#####

# Property to set JVM initial and maximum heap size
# when starting and stopping WebLogic Server.
curam.server.jvm.heap.size=1024
```

Figure 2: AppServer properties sample

Note:

1. The security.username used here in the *AppServer.properties* file should not be the same as any user that will exist as an application user.

With the implementation of strong password enforcement in WebLogic Server 11g Release 1, the Weblogic application server password must be at least eight alphanumeric characters in length with at least one number or a special character.

2. The **configure** target cannot be run while H2 database is in use. For more information on H2 database consult the *Cúram Development Environment Installation Guide*.
3. The *CuramMBean.jar* is copied from the *<SDEJ>/lib* directory to the *<WebLogic Server Install directory>/wlserver/server/lib/mbeantypes* directory by the **configure** target. The location of the *CuramMBean.jar* can be changed by manually copying to any location supported by the WebLogic Server. However, depending on the location, an extra parameter may be needed when executing the **startserver** command. For more details, please refer to [Start a WebLogic Server on page 20](#).
4. You have the option of setting *curam.principal.name* in the *AppServer.proerties* file. This defaults to Curam but if changed will update the principle-name in *weblogic.xml* and *weblogic-ejb-jar.xml* files.

Table 1:

| Element | Required Optional | Description |
|------------------|--|--|
| <principal-name> | Required if <externally-defined> is not defined. | Specifies the name of a principal that is defined in the security realm. You can use multiple <principal-name> elements to map principals to a role. For more information on security realms |

Configuring a web server plug-in in Oracle WebLogic Server

If a web server is configured in the topology, you must configure a web server plug-in in Oracle WebLogic Server. The web server plug-in enables Oracle WebLogic Server to communicate with the web server. For information about how to configure the web server's HTTP verb permissions to mitigate verb tampering, see .

About this task

To enable a web server plug-in in Oracle WebLogic Server, you can run the `configurewebserverplugin` target.

Procedure

To start automatic configuration, in the `SERVER_DIR` directory, run the `configurewebserverplugin` target, as shown in the following example command.

The `configurewebserverplugin` target requires a mandatory `server.name` argument that specifies the name of the server when the target is started.

```
build configurewebserverplugin -Dserver.name=CuramServer
```

What to do next

For information about how to configure the web server's HTTP verb permissions to mitigate verb tampering, see .

3.2 Security configuration for Oracle WebLogic Server

The default authentication mechanism in *WebLogic Server* is by authentication providers, and application security is implemented by using a custom authentication provider.

For more information about the default configuration, see .

The application and *WebLogic Server* support the use of alternative authentication mechanisms, such as an LDAP directory server or a single sign-on solution. WebLogic Server provides authentication providers that can be configured to work with LDAP directory servers and for single sign-on solutions, the third-party vendor often produces a custom authentication provider to work with *WebLogic Server* . Where an alternative provider is to be used for authentication, the Cúram authentication provider should verify only that users are valid for authorization purposes.

To configure the Cúram provider for identity-only authentication, you must set the `curam.security.check.identity.only` property to `true` in the `AppServer.properties` file before you run the **configure** target. You can configure additional authentication providers manually only after the `configure` target is run. For more information, see .

An optional property that enables logging for the Cúram authentication provider is available. Set the `curam.security.login.trace` to `true` to add tracing information to the *WebLogic Server* log file during the authentication process. You must set the property in the `AppServer.properties` file before you run the **configure** target.

3.3 Time Zone Configuration

If multiple server machines are used, they all must have their clocks in sync and be in the same time zone so that the natural ordering of date/times on the database accurately reflects the order that the events occurred in the real world.

For example, if on the database record *A* has a creation date/time field earlier than that on record *B*, then we can say for sure that *A* was created before *B*, no matter which server created either record.

The time zone of the server(s) must never change during the lifetime of the application. The reason for this is that the time zone assumed when storing dates in the database is the current server's time zone; therefore if the server's time zone changes then all dates entered prior to the time zone change will be out by the number of hours equal to the difference between the old and new time zones.

3.4 Starting and Stopping WebLogic Servers

For Windows platforms, the Cúram SDEJ provides Ant targets to aid in the starting and stopping of the WebLogic server.

These targets must be executed from the `<SERVER_DIR>` directory and, for the **configure** target, they require the `AppServer.properties` file to be set up correctly. For more information, see [3.1 WebLogic Server configuration on page 17](#). They also require a number of extra parameters to be specified. These are detailed below.

Start a WebLogic Server

The Ant target for starting a WebLogic server is:

```
build startserver
```

and requires the following option:

- `-Dserver.name`

The name of the server to be started.

The following parameter is optional and is only needed where the `CuramMBean.jar` file has been placed in a directory other than the default `<WebLogic Server Install directory>/wlserver/server/lib/mbeantypes` or the `/domaindir/lib/mbeantypes` directory, (where `domaindir` represents the location of your domain.)

- `-Dweblogic.alternateMbeanTypesDir`

The location of the `CuramMBean.jar`

```
build startserver -Dserver.name=CuramServer
```

Figure 3: Example of Usage

```
build startserver -Dserver.name=CuramServer -
Dweblogic.alternateMbeanTypesDir=C:\Location
```

Figure 4: Example of Usage with `alternateMbeanTypesDir` set

Important: Before starting the application server for the first time you must have run the **database** target followed by the **prepare.application.data** target. Failing to run this sequence will likely result in transaction timeouts during first login and a failure to initialize and access the application. Whenever the **database** target is rerun (e.g. in a development environment) the **prepare.application.data** target must also be rerun.

Stop a WebLogic Server

The Ant target for stopping a WebLogic server is:

```
build stopserver
```

and requires the following option:

- `-Dserver.name`

The name of the server to be stopped.

```
build stopserver -Dserver.name=CuramServer
```

Figure 5: Example of Usage

Restart a WebLogic Server

The Ant target for restarting a WebLogic server is:

```
build restartserver
```

and requires the following option:

- `-Dserver.name`

The name of the server to be restarted.

The following parameter is optional and is only needed where the `CuramMBean.jar` file has been placed in a directory other than the default `<WebLogic Server Install directory>/wlserver/server/lib/mbeantypes` or the `/domaindir/lib/mbeantypes` directory, (where `domaindir` represents the location of your domain.)

- `-Dweblogic.alternateMbeanTypesDir`

The location of the `CuramMBean.jar`

```
build restartserver -Dserver.name=CuramServer
```

Figure 6: Example of Usage

```
build startserver -Dserver.name=CuramServer -
Dweblogic.alternateMbeanTypesDir=C:\location
```

Figure 7: Example of Usage with `alternateMbeanTypesDir` set

Note: If the server is not already started when attempting to restart it, the stop portion of the Ant target will not cause the target to fail.

Server Logging

The servers are started and stopped as Windows Services and can be found in the Service Listing in the format `<domain_name>_<server_name>`. The default server logging from this service gets placed in the file `<WebLogic Install directory>/<domain_repository>/<domain_name>/<node.name>/servers/<server.name>/logs/<server.name>_Redirect.log`. This log gets cleared upon server start-up.

4 Deployment

The final step after packaging the Cúram application and web services `.ear` files is to deploy them on the application server.

The default installation for the application and web services `.ear` is to deploy them on the server in the same installation of Oracle WebLogic Server.

4.1 Deployment

Before deploying an application, restart (or start) the WebLogic server as detailed in [3.4 Starting and Stopping WebLogic Servers on page 20](#). When deploying the Cúram application ensure that the database is configured correctly. If the database does not contain the necessary information deployment may fail for security/validation reasons.

The SDEJ provides Ant targets for deploying and undeploying applications on a WebLogic server. As with the `startserver/stopserver` targets, the `installapp/uninstallapp` targets require the `AppServer.properties` file to be configured correctly (see [3.1 WebLogic Server configuration on page 17](#)). The targets also require a number of options detailed in the following sections.

Deploy an Application

The Ant target to deploy or install an application (in the form of an `.ear` file) is:

```
build installapp
```

and requires the following options:

- `-Dserver.name`
The name of the server to install the application on.
- `-Dear.file`
The fully qualified name of the `.ear` file to install.
- `-Dapplication.name`
The name to identify the application when it is installed.

```
build installapp -Dserver.name=CuramServer -Dear.file=
$SERVER_DIR/build/ear/Curam.ear -Dapplication.name=Curam
```

Figure 8: Example of Usage

Note: The EAR file containing the server module must be deployed before installing any other (client-only) EAR files.

Change SYSTEM Username

It is strongly recommended that you change this username after deploying the application using the WebLogic Server administration console. The `Run As User` property should be changed from `SYSTEM` to the user of choice. The password of this user does not matter, since no authentication and only identity assertion is performed by WebLogic Server for JMS invocations.

The `Run As Principal Name` property can be found on the Configuration Tab for each of the MDB (Message Driven Bean) EJB modules deployed, and should match this value. This also requires updating `Run As User` property of `myrealmCuramAuthenticator` (Curam Authentication Provider) to new user of choice under `Security Realms` section in the WebLogic Server administration console. Consult the WebLogic Server documentation for more information on the usage of the administration console.

Note, if the username is changed, the new username must exist in the Users database table and this user must have a role of 'SUPERROLE'.

The `SYSTEM` user is the user under which JMS messages are executed.

Undeploy an Application

The Ant target to undeploy or uninstall an application is:

```
build uninstallapp
```

and requires the following options:

- `-Dserver.name`

The name of the server the application is installed on.

- `-Dapplication.name`

The name of the application to uninstall (as configured during install).

```
build uninstallapp -Dserver.name=CuramServer
-Dapplication.name=Curam
```

Figure 9: Example of Usage

4.2 Pre-compiling JSPs

During deployment, use the Ant `precompilejsp` target to pre-compile the JSPs of a client EAR before installing the EAR file. Pre-compiling the JSPs before installation speeds up the display of a page in the web browser the first time it is accessed.

The options for the `precompilejsp` target are:

- `-Dear.file`

The fully qualified name of the `.ear` file to be pre-compiled.

```
build precompilejsp -Dear.file=$SERVER_DIR/build/ear/WLS/
Curam.ear
```

Figure 10: Example of Usage

This target will overwrite the `<Curam.ear>` with a copy that contains the pre-compiled JSP s and can then be deployed as described in [4.1 Deployment on page 23](#).

4.3 Testing Deployment

When the application is installed on a configured WebLogic Server installation the application should be started and tested.

Note: The installation of a web services application may also be required.

To do this, ensure the relevant server is started (there is no need to restart the server after an application is deployed), and open the following page in a web browser:

```
https://<some.machine.com>:<port>/<context-root>
```

where,

`<some.machine.com>` identifies the the host name or IP address where your WebLogic Server system is running, `<port>` identifies the SSL port of the server the application is deployed on and `<context-root>` identifies the Context Root of the WAR module (see [2.3 Multiple EAR files on page 15](#), for details). The SSL port is one number up from the `curam.server.port` specified in the `AppServer.properties` file.

Before the page can be opened, the browser will be directed to the login page. Log-in with a valid application username and password and the browser will be redirected to the requested page.

Note: The usage of EAR file name `Curam.ear` for option-`Dear.file` and usage of application server name `Curam` for option-`Dapplication.name` in the examples of this chapter are for illustrative purposes. Based on your customized application and deployment strategy these values may change.

5 Manual WebLogic Server configuration

If required, the Oracle WebLogic Server installation can be configured manually. However, manual configuration is not recommended. For informational purposes only, the following manual steps are required to configure and deploy on WebLogic Server on Microsoft™ Windows™.

5.1 Creating a WebLogic Server Domain

The Domain Configuration Wizard is a tool to aid in the configuration of a WebLogic Server domain.

On Windows Platforms the domain configuration wizard can be invoked from:

Programs > Oracle WebLogic > WebLogic Server 11gR1 > Tools > Configuration Wizard

On UNIX Platforms the domain configuration wizard can be invoked by running:

```
<wls.home>/weblogic/common/bin/config.sh
```

where <wls.home> is the WebLogic installation directory (e.g. /opt/wls).

Domain Configuration

About this task

When the domain configuration wizard opens, follow the steps below, exactly as outlined, to configure the domain:

Procedure

1. Select **Create a new WebLogic Domain** and click **Next**;
2. Select **Generate a domain configured automatically to support the following products**. Ensure the **WebLogic Server** checkbox is selected and click **Next**;
3. Set the **Domain name**.

The **Domain name** (and all server names) can be called anything so long as the *AppServer.properties* file is updated correctly. Leave **Domain location** as default and click **Next**;

4. Configure an Administrative Username and Password. Note that the password must be at least eight alphanumeric characters with at least one number or a special character; for example: “weblogic” (username) and “weblogic1” (password). Confirm the password and click **Next**;
5. Choose the **Startup mode** and **Java SDK** as necessary and click **Next**;
6. Choose nothing in the **Select Optional Configuration** screen and click **Next**;
7. Review the Configuration settings and click **Create** to construct the domain.
8. The Configuration Wizard is now complete. Click **Done** to end the Configuration Wizard and exit the installer.

The Cúram application requires a `MaxPermSize` of at least 256m to run. This must be configured as a parameter to the newly created domain. Edit the file relevant to your platform (located at `<WebLogic Install directory>/<domain_repository>/<DomainName>/bin/`), either:

- a) `setDomainEnv.cmd` - Windows
- b) `setDomainEnv.sh` - UNIX

and where it passes `MaxPermSize`, set this to 256m.

Note: The setting of 256m is for illustrative purposes. The optimum value should be determined by monitoring the memory utilized for your server.

5.2 Java Virtual Machine Configuration

The Cúram application requires additional parameters to be passed to the JVM running the Application Server. Set the following system environment variables:

- `USER_MEM_ARGS` : This should be set to a value relevant to your application e.g. -Xmx712m. This environment variable is set by the `app_runtimewls.xml` script using these properties, which can be defaulted as follows or set in the `AppServer.properties` properties file:

Table 2: Memory Arguments

| Property Name | Default Value | Description |
|--|--|---|
| <code>curam.server.jvm.heap.size</code> | 1024 | Specifies the JVM heap initial and maximum sizes. |
| <code>curam.server.jvm.permgen.size</code> | <code>-XX:PermSize=128m - XX:MaxPermSize=256m</code> | Specifies the initial and maximum PermGen sizes. |

If you override the `USER_MEM_ARGS` environment variable directly you must also provide values for the JVM PermGen space as shown in [Table 2: Memory Arguments on page 28](#).

- `WLS_REDIRECT_LOG` : Output file e.g. `<WebLogic Server Install directory>/<domain_repository>/<DomainName>/logs/<server.name>_Redirect.log`
- `JAVA_OPTIONS` : This environment variable is used for two purposes.
 1. This environment variable should be used to set the headless mode. The headless mode property is only required for UNIX and it should have a value of `-Djava.awt.headless=true`
 2. This can be used to pass additional parameters to the WebLogic Server JVM when starting the Application Server.

5.3 Set up Database Driver

The version of the Oracle® Database driver file, *ojdbc6.jar*, used by WebLogic Server is not the same as that shipped with the application product, under the *drivers* directory of the SDEJ installation.

Follow the step below to set up WebLogic Server to use Oracle **Database Driver** shipped with the application product:

- Replace the *ojdbc6.jar* file in *WLS_HOME\lib* folder with one shipped with the application product, under the *drivers* directory of the SDEJ installation, e.g. *D:\Curam\SDEJ\drivers*;

where, the *WLS_HOME* variable points to the */server* directory of the WebLogic Server installation, for example: *d:\WLS\weblogic\server* or */opt/wls/weblogic/server*.

Note: The copies of Oracle Thin drivers installed with WebLogic Server and other supporting files are installed in *WLS_HOME\ext\jdbc* directory. There is a subdirectory in this folder for each DBMS. If you need to revert to the version of the driver installed with WebLogic Server at any point, then you can copy the file from *WLS_HOME\ext\jdbc\oracle\11g* to *WLS_HOME\lib* folder.

5.4 Starting the WebLogic Administration Service

Procedure

1. Open a command prompt and navigate to *<WebLogic Install directory>/<domain_repository>/<DomainName>* directory, (e.g. *C:/Weblogic/user_projects/domains/<domain_name>*). Execute the command **startWeblogic** from this directory. When the server has started correctly the following line should be displayed: *<Server started in RUNNING mode>* in the log file, which was configured previously in [5.2 Java Virtual Machine Configuration on page 28](#) or in the command prompt if it's not already configured.
2. To open the Administration Console, the following URL should be opened in a web browser:
http://<IP Address>:7001/console
3. Login with the username and password configured during installation (e.g. “weblogic” / “weblogic1”).

5.5 Database Configuration

About this task

Open the Administration Console as detailed in the previous section.

Procedure

1. Navigate to **<DomainName> > Services > JDBC > Data Sources**;
2. Click the **New** button;
3. Enter the following fields:

Name: “curamdb”

JNDI Name: “jdbc/curamdb”

Change the **Database Type** to be “Oracle”

4. Click the **Next** button
5. Set the **Database Driver** to be “Oracles Driver (Thin XA) for Instance connections; Versions:9.0.1,9.2.0,10,11”;
6. Click the **Next** button
7. Leave the default for **Transaction Options** and click the **Next** button.
8. Set the following fields:

Database Name : This setting depends on how you want to connect to the Oracle database; i.e., either using the Oracle service name or Oracle SID name.

Set this value to the value of `curam.db.oracle.servicename` in `<SERVER_DIR>/project/properties/Bootstrap.properties` to connect to database using the Oracle service name, e.g..

If you want to connect to an Oracle database using the SID name, then set this value to the value of `curam.db.name` in `<SERVER_DIR>/project/properties/Bootstrap.properties` e.g. “yourhost”.

Hostname : Set this value to the value of `curam.db.servername` in `<SERVER_DIR>/project/properties/Bootstrap.properties`, e.g. “gonzo.<host_name>”.

Port : Set this value to the value of `curam.db.serverport` in `<SERVER_DIR>/project/properties/Bootstrap.properties`, e.g. “1521”.

Database Username: Set this value to the value of `curam.db.username` in `<SERVER_DIR>/project/properties/Bootstrap.properties`, e.g. “curam”.

Password: Set this value to the value of `curam.db.password` in `<SERVER_DIR>/project/properties/Bootstrap.properties`. Note that password in the `Bootstrap.properties` property file is encrypted, and you need to set the unencrypted, plain-text version of this password here.

Confirm Password : confirm the entered password.

9. Click the **Next** button
10. Here if you are connecting using Oracle service name, then change the **URL** value as shown below before testing the configuration:

```
jdbc:oracle:thin:@//serverName:port/databaseServiceName
```

Where `serverName` is the name of the server hosting the database.

Where `port` is the port number the database is listening on.

Where *databaseServiceName* is the service name of the database.

Leave all other fields untouched unless a specific change is required.

Click the **Test Configuration** button to test settings.

11. Click the **Next** button;
12. Review the settings and click the **Next** button;
13. Select **AdminServer** as the target server;

For manual configuration you will need to set the value of `curam.server.name` property in `AppServer.properties` file to be `AdminServer`.

14. Click the **Finish** button;

It is a good idea to restart the AdminServer at this point, to ensure the changes are correct. To do this:

- a) Navigate to **<DomainName> > Environment > Servers**;
- b) Select the **Control** tab, then select AdminServer in the Server's list and click **Shutdown > When work completes**;
- c) Click the **Yes** button to shutdown the AdminServer.

Although it is very easy to kill managed servers by using Ctrl + C when it is running in a command prompt, never do this. Always use the Administration Console to shutdown all managed servers as described above. Using Ctrl + C from a Command Prompt will significantly slow down a machine after a couple of restarts. This is because memory is not released as it should be and the only remedy is to restart the machine.

5.6 Setup Security

About this task

Copy the `CuramMBean.jar` from the `<SDEJ>/lib` directory to the `<WebLogic Server Install directory>/wlserver/server/lib/mbeantypes` directory.

This also requires to copy the `CryptoConfig.jar` to the jre used by this WebLogic Server Installation, within the `java/jre/lib/ext` directory. This is same for any other WebLogic Server Installation Setup.

Restart the AdminServer and start the Administration Console as described in the previous section.

Procedure

1. Navigate to **<DomainName> > Security Realms**;
2. Click on **myrealm** in the **Realms** list;
3. Click on **Providers** tab;
4. Click on **Authentication** tab;
5. Click the **New** button;
6. Enter the following fields:

Name : “myrealmCuramAuthenticator”

Type : “CuramAuthenticator”

7. Click the **OK** button;
8. In the list of **Authentication Providers**, click the **DefaultAuthenticator** checkbox;
9. Click the **Delete** button;
10. Click on **myrealmCuramAuthenticator** in the **Authentication Providers** list;
11. Ensure the **Control Flag** value is set to “REQUIRED”.

If not change the value to “REQUIRED”;

12. Click the **Save** button;
13. Select the **Provider Specific** tab. This tab contains settings for configuring Cúram security in WebLogic Server. The defaults should not be changed unless you wish to modify the security configuration. [Table 3: Provider Specific Options on page 32](#) explains the details of the various options.

You must enter the digested password for the Admin Password value. Generate this password by running the supplied Ant digest target; e.g., `ant digest -Dpassword=weblogic1`.

If any changes are made click the **Save** button;

14. Click the **Save** button; ensure that there are no errors.

Table 3: Provider Specific Options

| Field | Description |
|---------------------|--|
| Check Identity Only | Optional. If this box is checked the authentication provider will not perform the usual authentication verifications. Instead it will simply ensure that the user exists on the database table. This option is intended where LDAP support is required or an alternative authentication mechanism is to be used. |
| Admin Username | Required. This is the username of the WebLogic Server administration user. This user is excluded from Cúram authentication. |
| Admin Password | Required. This is the encrypted password of the WebLogic Server administration user. Generate the encrypted password by running the supplied Ant digest target; e.g., <code>ant digest -Dpassword=weblogic1</code> |
| Port | Required. This is the port of the machine on which the Cúram application will run. The default is 7003. In a clustered environment this should be set to a ',' separated list of ports to support multiple servers. |
| Login Trace | Optional. This box should be checked to debug the authentication process. If selected the invocation of the Cúram authentication provider will result in tracing information being added to the WebLogic Server log file. |

| Field | Description |
|-------------|---|
| Run As User | Required. See section Change SYSTEM Username on page 24 for a description of this property. The default is SYSTEM. |
| Hostname | Required. This is the hostname of the machine on which the Cúram application will run. The default is localhost. In a clustered environment this should be set to a ',' separated list of host names to support multiple servers. |

Note: While configuring the Cúram Authenticator Provider in a clustered environment, the ordering of `hostname` and `port` attributes is important. There is a one to one mapping between the servers and ports specified. For example:

```
Port=7001,7003,7005
Hostname=host1,host2,host3
```

Here `host1` is running the WebLogic Server on port 7001 and `host3` 7005 is running the WebLogic Server on port

You must enable SSL support. To do this:

- Navigate to `<DomainName> > Environment > Servers`;
- Select the AdminServer from the list of servers.
- From the **General** tab click the **SSL Listen Port Enabled** checkbox;
- Click the **Save** button;
- Restart your server to take changes.

5.7 Setup Transport Layer Security

About this task

This task is used for setting the Transport Layer Security (TLS) protocol to TLS v1.2.

Note : TLS v1.2 is only available Java version 7.0 or higher.

Procedure

- Navigate to `<WebLogic Install directory>/<domain_repository>/<domain_name>/bin` directory, (e.g. `C:/Weblogic/user_projects/domains/<domain_name>/bin`).
- Edit the file named `setDomainEnv.cmd` on a Microsoft Windows platform or named `setDomainEnv.sh` on non Microsoft Windows platforms;
- On a Microsoft Windows platform, find the last instance of **set JAVA_OPTIONS=%JAVA_OPTIONS%** in the file `setDomainEnv.cmd` and replace it with **set JAVA_OPTIONS=%JAVA_OPTIONS% -Dweblogic.security.SSL.minimumProtocolVersion=TLSv1.2**
- On non Microsoft Windows platforms, find the last instance of **JAVA_OPTIONS="{JAVA_OPTIONS}"** in the file `setDomainEnv.sh`

and replace it with **JAVA_OPTIONS="{JAVA_OPTIONS} - Dweblogic.security.SSL.minimumProtocolVersion=TLSv1.2"**

Note: Oracle WebLogic Server automatically selects a set of strong ciphers for TLS1.2 communication. Over time, strong ciphers might be re-categorized as weak ciphers. Therefore, it is recommended that you review the selected ciphers regularly to ensure that they are up to date. For more information about how to review and reconfigure the ciphers, see the relevant [WebLogic Server documentation](#).

5. Save the file and exit.

5.8 Setup JMS

About this task

The Cúram application uses persistent messages. A JMS file store must be created for storing persistent messages. To create a directory on the file system where the JMS file store will be kept (e.g. *<WebLogic Server Install directory>/user_projects/domains/<DomainName>/jms_file_store*), complete the following steps using the Administration Console:

Procedure

1. Navigate to **<DomainName> > Services > Persistent Stores**;
2. Click **New > Create FileStore** and set the following properties:
 - Name** : "CuramJMSFileStore"
 - Target** : "AdminServer"
 - Directory** : <directory created above>;
3. Click the **OK** button;
4. Navigate to **<DomainName> > Services > Messaging > JMS Servers**;
5. Click the **New** button to configure a new JMSServer and set the following properties:
 - Name** : "CuramJMSServer"
 - Persistent Store** : "CuramJMSFileStore"
6. Click the **Next** button and select the AdminServer as the target server;
7. Click the **Finish** button to complete;

5.9 Queue Setup

Procedure

1. Navigate to **<DomainName> > Services > Messaging > JMS Modules**;
2. Click the **New** button and enter the following details:

Name : “op-jms”

Descriptor File Name : “jms/op-jms.xml”

3. Click the **Next** button and check the **AdminServer** checkbox as the target server;
4. Click the **Next** button;
5. Click the **Would you like to add resources?** checkbox and click the **Finish** button to complete;

There are four regular queues and two error queues that must be configured. The error queues must be configured first. The following setup should be repeated, replacing <QueueName> with each of the following queues (in the order listed): CuramDeadMessageQueue, DPError, WorkflowError, DPEnactment, WorkflowEnactment, and WorkflowActivity.

6. Navigate to <DomainName> > **Services** > **Messaging** > **JMS Modules**;
7. Click on **op-jms** in **JMS Modules** list;
8. Click the **New** button inside the **Configuration** tab;
9. Select **Type** “Quota” and click the **Next** button;
10. Enter the following details:

Name : “<QueueName>.Quota”

Leave the default for the **Bytes Maximum** : “9223372036854775807”

Leave the default for the **Messages Maximum** : “9223372036854775807”

Leave the default for the **Policy** : “FIFO”

Leave the default for the **Shared** : “False”.

11. Click the **OK** button;
12. Click the **New** button and select **Type** “Queue”;
13. Click the **Next** button and enter the following details:

Name : “<QueueName>”

JNDI Name : “jms/<QueueName>”.

14. Click the **Next** button;
15. If the **SubDeployments** drop-down list is empty click on **Create a New SubDeployment** and enter **SubDeployment Name** : “CuramJMSServer”;
16. Click the **OK** button;
17. Set **SubDeployment** : “CuramJMSServer”
18. Select “CuramJMSServer” as the target JMS Server;
19. Click the **Finish** button;
20. Click the “<QueueName>” just configured.
21. Click on **Thresholds and Quotas** tab;
22. Set the **Quota** : “<QueueName>.Quota”;
23. Click the **Save** button;
24. Select the **Overrides** tab and set the **Delivery Mode Override** to “Persistent”. Click **Save**;

25. Select the **Delivery Failure** tab and set the **Redelivery Limit** to “1”. Set the **Error Destination** to “none” for “CuramDeadMessageQueue”, “CuramDeadMessageQueue” for “DPError” and “WorkflowError”, “DPError” for “DPEnactment”, and “WorkflowError” for “WorkflowEnactment” and “WorkflowActivity”.
26. Click the **Save** button;

5.10 Queue Connection Factory Setup

About this task

To configure the XA Queue Connection Factory, complete the following steps:

Procedure

1. Navigate to <DomainName> > **Services** > **Messaging** > **JMS Modules**;
2. Click on **op-jms** in **JMS Modules** list;
3. Click the **New** button inside the **Configuration** tab;
4. Select **Type** “Connection Factory” and click **Next**;
5. Set the following fields:
 - Name** : “CuramQueueConnectionFactory”
 - JNDIName** : “jms/CuramQueueConnectionFactory”;
6. Click the **Next** button;
7. Click the **Finish** button;
8. Click the “CuramQueueConnectionFactory” just configured;
9. Select the **Configuration** tab and then the **Transactions** sub-tab. Ensure that **XAConnection Factory Enabled** is selected. Click the **Save** button.

To configure the non-XA Queue Connection Factory, complete the following steps:

10. Navigate to <DomainName> > **Services** > **Messaging** > **JMS Modules**;
11. Click on **op-jms** in **JMS Modules** list;
12. Click the **New** button inside the **Configuration** tab;
13. Select **Type** “Connection Factory” and click **Next**;
14. Set the following fields:
 - Name** : “CuramQueueConnectionFactoryNonXA”
 - JNDI Name** : “jms/CuramQueueConnectionFactoryNonXA”;
15. Click the **Next** button;
16. Click the **Advanced Targeting** button;
17. If the **SubDeployments** drop-down list is empty click on **Create a New SubDeployment** and enter **SubDeployment Name** : “CuramJMSServer”.
18. Click the **OK** button;
19. Set **SubDeployment** : “CuramJMSServer”

20. Select “CuramJMSServer” as the target JMS Server;
21. Click the **Finish** button;
22. Click the “CuramQueueConnectionFactoryNonXA” just configured;
23. Select the **Configuration** tab and then the **Transactions** sub-tab. Ensure that **XAConnection Factory Enabled** is not selected. Click the **Save** button.

5.11 Topic Setup

About this task

To support cache reloading in the Cúram application, a topic must be configured as follows:

Procedure

1. Navigate to <DomainName> > **Services** > **Messaging** > **JMS Modules**;
2. Click on **op-jms** in **JMS Modules** list;
3. Click the **New** button inside the **Configuration** tab;
4. Select **Type** “Quota” and click **Next**;
5. Enter the following details:
 - Name** : “CuramCacheInvalidationTopic.Quota”
 - Bytes Maximum** : Leave the default “9223372036854775807”
 - Messages Maximum** : Leave the default “9223372036854775807”
 - Policy** : Leave the default “FIFO”
 - Shared** : Leave the default “False”.
6. Click the **OK** button,
7. Click the **New** button and select **Type** “Topic”;
8. Click **Next** and enter the following details:
 - Name** : “CuramCacheInvalidationTopic”
 - JNDI Name** : “jms/CuramCacheInvalidationTopic”.
9. Click the **Next** button;
10. If the **SubDeployments** drop-down list is empty click on **Create a New SubDeployment** and enter **SubDeployment Name** : “CuramJMSServer”.
11. Click the **Finish** button;
12. Set **SubDeployment** : “CuramJMSServer”
13. Select “CuramJMSServer” as the target JMS Server;
14. Click the **Finish** button;
15. Click the “CuramCacheInvalidationTopic” just configured.
16. Click on **Thresholds and Quotas** tab;
17. Set the **Quota** : “CuramCacheInvalidationTopic.Quota”;

18. Click the **Save** button;
19. Select the **Delivery Failure** tab and set the **Redelivery Limit** to “1”. Ensure the **Error Destination** is set to (none) and click the **Save** button;

5.12 Topic Connection Factory Setup

Procedure

1. Navigate to <DomainName> > **Services** > **Messaging** > **JMS Modules**;
2. Click on **op-jms** in **JMS Modules** list;
3. Click the **New** button inside the **Configuration** tab;
4. Select **Type** “Connection Factory” and click **Next**;
5. Set the following fields:
 - Name** : “CuramTopicConnectionFactory”
 - JNDIName** : “jms/CuramTopicConnectionFactory”;
6. Click the **Next** button
7. Click the **Advanced Targeting** button;
8. If the **SubDeployments** drop-down list is empty click on **Create a New SubDeployment** and enter **SubDeployment Name** : “CuramJMSServer”.
9. Click the **OK** button;
10. Set **SubDeployment** : “CuramJMSServer”
11. Select “CuramJMSServer” as the target JMS Server;
12. Click the **Finish** button;
13. Click the “CuramTopicConnectionFactory” just configured;
14. Select the **Configuration** tab and then the **Transactions** sub-tab. Ensure that **XAConnection Factory Enabled** is selected. Click the **Save** button.

5.13 Manual Deployment

About this task

It is possible to manually deploy an `.ear` file using the Administration Console. For this to succeed, the relevant server must be started. In the Administration Console, complete the following steps:

Procedure

1. Navigate to <DomainName> > **Deployments**;
2. Click **Install**.
3. Navigate to the location of the `.ear` file.

The default location for the server `.ear` is:

%SERVER_DIR%/build/ear/WLS/Curam.ear

4. Select the *.ear* file from the list and click the **Next** button.
5. Accept defaults and click the **Next** button.
6. Accept the defaults and click the **Finish** button.
7. Select the application just deployed in the **Deployments** list and click **Start > Servicing all requests**.
8. Click the **Yes** button;
9. Finally, test the application deployment.

For example, point a Web browser at the URL for the deployed application e.g. *https://localhost:7002/Curam*.

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

Any references in this information to non-Merative websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those

websites are not part of the materials for this Merative product and use of those websites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Merative, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at <https://www.merative.com/privacy>.

Trademarks

Merative™ and the Merative™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.