merative™

Cúram 8.1.3

**Deploying on IBM® WebSphere®
Application Server Guide**

Note

Before using this information and the product it supports, read the information in Notices on page 55

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

Notices... 55

1 Deploying on IBM® WebSphere® Application Server

To deploy Cúram on IBM® WebSphere® Application Server, you must configure the server and deploy the application EAR files.

This guide shows you how to build Cúram for deployment on the base version of WebSphere® Application Server. The guide also details the support provided for configuring and deploying on WebSphere® Application Server.

> **Note:** For information on using the application with the Network Deployment edition of WebSphere® Application Server consult 1.8 Manual WebSphere® Application Server configuration on page 26.

It is a prerequisite that the reader have knowledge of the Cúram development environment. In other words, that they know how to develop and build a server application and web client. The guide also presumes that WebSphere Application Server is installed.

1.1 Building EAR files

Package Cúram into EAR (Enterprise ARchive) files.

The main step before deployment of a Cúram application is to package it into EAR (Enterprise ARchive) files. The server application (which incorporates the web client and server) and web services application are packaged into separate EAR files, and the Server Development Environment (SDEJ) provides build targets that perform this task.

Before the targets described in the following section are executed, ensure that the *WAS_HOME* environment variable is set.

This should point to the base directory of the base WebSphere Application Server installation. For example: `d:\WebSphere\AppServer` or `/opt/WebSphere/AppServer`.

The enterprise application EAR file

Create the application EAR file for IBM® WebSphere® Application Server

Building the application EAR file

Execute the following target from the root directory of the server project to create the application EAR file for WebSphere® Application Server:

`build websphereEAR`

This target creates a ready to install EAR file, `<SERVER_MODEL_NAME>.ear` located in `<SERVER_DIR>/build/ear/WAS`

The environment variables *SERVER_MODEL_NAME* and *SERVER_DIR* specify the name of the model in the project and the root directory of the project respectively. Before executing this

target, a fully built application must be available. For more information, see the *Cúram Server Developer's Guide*.

> **Note:** An EAR file cannot be built for H2 database For more information see the *Cúram Development Environment Installation Guide*.

The **websphereEAR** 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, that is, 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 Cúram 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 has the following structure and contents:

- *META-INF* Directory :

  - *application.xml*: A generated file that lists the mapping of EJB modules to JAR files that are contained in the application.
  - *ibm-application-bnd.xmi*: A generated WebSphere Application Server-specific extension file.
  - *ibm-application-ext.xmi*: A generated WebSphere Application Server-specific extension file.
  - *was.policy*: A security policy file that grants the application the Java™ permission java.security.AllPermission.
  - *MANIFEST.MF*: The manifest file that 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: *antlr.jar*, *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*: 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.
- *ibm-ejb-jar-bnd.xmi*: A generated WebSphere Application Server -specific extension file.
- *ibm-ejb-jar-ext.xmi*: A generated WebSphere Application Server -specific extension file.
- *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 *application.jar*, *codetable.jar*, *events.jar*, *struct.jar*, *messages.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.

The web services application

Automatically generate Web Service Definition Language (WSDL) defined web services. Application developers can 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
Build the web services EAR file.

The EAR file

Execute the following target from the root directory of the project to create the EAR file for web services:

build websphereWebServices

Optional overrides are:

- prp.webipaddress is the IP address on which the server hosting the web services is listening. The default is *http://localhost:2809*.
- prp.contextproviderurl is the URL of the JNDI context provider. This is the address of the server that hosts the Cúram components being made accessible though web services. The default is *iiop://localhost:2809*.
- prp.contextfactoryname is the JNDI context factory name. The default for this is com.ibm.websphere.naming.WsnInitialContextFactory and should rarely need to be changed.

This target creates a ready to install EAR file, *<SERVER_MODEL_NAME>WebServices.ear* located in *<SERVER_DIR>/build/ear/WAS*.

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

Under the hood

The **websphereWebServices** 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 components* 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 is treated as if it also had a stereotype of ejb. This is because Web Service interfaces are wrappers on publicly available BPOs.

> **Note:** 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 contains a `META-INF` directory and a web service `WAR` file.

The 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.

  - `ibm-application-bnd.xmi`

    A generated WAS specific extension file.

  - `ibm-application-ext.xmi`

    A generated WAS specific extension file.

  - `was.policy`

    WAS security policy file that grants the application the Java permissionjava.security.AllPermission.

  - `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 Axis web services engine.

  - `appwebservices.jar`

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

- *server-config.wsdd*

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

Web service WSDL

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

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

http://localhost:9082/CuramWS/services/MyTestService

the WSDL description is at the URL:

http://localhost:9082/CuramWS/services/MyTestService?wsdl

The URL

http://localhost:9082/CuramWS/services returns a web page that lists all web services deployed and a link to their WSDL files.

The general URL format of the locations above is as follows: *http://<web-server>:<port-number>/<ServerModelName>WS/services/<BPO-name>*.

Multiple EAR files

Building an Application EAR also takes an optional file to allow you to split the client components into different WAR and EAR files and also to allow control of some of the EAR configuration and included modules.

The optional 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>
```

Each file can have multiple `ear` elements and results in an EAR file being produced in the *SERVER_DIR/build/ear/WAS* 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.

You can 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 (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 Cúram application must be available. For details on how to build an application, see the *Cúram Server Developer's Guide*.

Alternative targets

The **websphereEAR** target builds an Cúram application `.ear` file that contains both the web client and the application. You can also build an application `.ear` file that contains only the web application or only the server application.

You might need these configurations where the web client and server application need to be installed on separate servers. For example, to support secure access to the Cúram application for external users a new web client application can be developed. This web application can be deployed on its own and use an existing server application. For more information about External Access Security, see the [Server Developer's Guide](#) Server Developer's Guide.

To build a `.ear` file that contains only the web client application, enter the following command:

```
build websphereEAR -Dclient.only=true
```

To build a `.ear` file that contains only the server application, enter the following command:

```
build websphereEAR -Dserver.only=true
```

1.2 Configuring WebSphere® Application Server

Configuring IBM® WebSphere® Application Server involves setting up a profile, data source, a number of servers and configuring the JMS and security settings. Execute the **configure** target provided by the SDEJ to perform these tasks.

Configuring WebSphere® Application Server is similar on all platforms and the Server Development Environment (SDEJ) provides a number of Ant targets to aid the configuration and management of the installation.

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

> **Note:** The **configure** target overwrites the *default* profile created by WebSphere® Application Server unless *-Dkeep.profile=true* is passed to the target.

The profile created by the **configure** target takes the following defaults unless specifically overridden when calling the target.

- profile.name=AppSvr01
- cell.name=${node.name}Cell

Execute the command **build configure** from the `<SERVER_DIR>` directory to invoke automatic configuration. The files `AppServer.properties` and `Bootstrap.properties` must be in the `<SERVER_DIR>/project/properties` directory. You can override this default location for the properties file by specifying **-Dprop.file.location=<new location>** when executing the **configure** target. See the *Cúram Server Developer's Guide* for more information on the setup of a

Bootstrap.properties. [1.2 Configuring WebSphere® Application Server on page 15](#) shows example contents of the *AppServer.properties* file.

```
## APPLICATION SERVER PROPERTIES

# Property to indicate WebSphere is installed.
as.vendor=IBM

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

# The name of the WebSphere Node
node.name=MyNode

# The name of the server on which the application will be hosted.
curam.server.name=CuramServer
curam.server.port=2809

####################################################
## THE FOLLOWING PROPERTIES ARE FOR WebSphere ONLY ##
####################################################
# The alias that should be used for the database authorization
curam.db.auth.alias=databaseAlias

# HTTP Port for the server on which the client
# will be accessed
curam.client.httpport=9044

# HTTP Port for the server on which the Web services
# will be accessed
curam.webservices.httpport=9082

# Property to set JVM initial and maximum heap size.
curam.server.jvm.heap.size=1024
```

By default the **configure** target establishes a Type 4 Universal Driver (XA) data source. However, you may configure a Type 2 Universal Driver (XA) data source by setting the curam.db.type2.required property in the *AppServer.properties* file.

Also by default, the **configure** target sets the JVM initial and maximum heap size to "1024" MB. However, you can override the default JVM initial and maximum heap size by setting the curam.server.jvm.heap.size property in the *AppServer.properties* file.

> **Note:**
>
> 1. The setting of the Java heap as described in the [1.2 Configuring WebSphere® Application Server on page 15](#) example and set by the configuration scripts is for illustrative purposes. Based on the size of your customized application, deployment strategy, etc. these settings may be too low or too high. The optimum value should be determined by monitoring the memory performance of your server.
> 2. Memory issues have been noticed with the WebSphere Application Server wrapped database drivers during the retrieval of large CLOBs and BLOBs (3MB+) from the database. These issues may be worked-around by increasing the Max Heap Size JVM parameter as appropriate on the deployed server.
> 3. The **configure** target cannot be run when H2 database is in use. For more information on H2 database consult the *Cúram Development Environment Installation Guide*.

Configuring a web server plug-in

If a web server is configured in the topology, you must configure a web server plug-in in WebSphere® Application Server. For information about how to configure the web server's HTTP verb permissions to mitigate verb tampering, see [Enabling HTTP verb permissions](#).

About this task
Complete the following tasks:

- Add the web server virtual hosts to the client hosts configuration in WebSphere® Application Server.
- Propagate the plug-in key ring for the web server.
- Map the modules of any deployed applications to the web server.

To complete the previous tasks, you can run the `configurewebserverplugin` target. The target configures the WebSphere® Application Server only when web servers are configured in the topology. You can run the target even after the applications are deployed or redeployed. In such cases, the target maps only the modules of any redeployed applications to the web server.

> **Note:** Do not use the `configurewebserverplugin` target in a z/OS environment because the plug-in configuration differs.

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 .

1.3 Configuring security

The default security configuration of Cúram involves the default file-based user registry and a JAAS Login Module. For more information, see the *Default Configuration for IBM WebSphere Application Server* section in the *Cúram Security Handbook*.

There are a number of alternative security configurations that can be used with WebSphere Application Server. The configurations are available to support the use of alternative authentication mechanisms, such as an LDAP directory server or a single sign-on solution.

To avail of a different configuration the properties detailed in the following sections should be set in the *AppServer.properties* file before running the `configure` target. Any alternative authentication mechanisms should be configured manually after running the `configure` target with the relevant properties set. To configure the login module for identity only authentication

the curam.security.check.identity.only property should be set to `true`.This is to ensure that the configured alternative authentication mechanism is used.

The *Identity Only Authentication* section in the *Cúram Security Handbook* should be consulted for further details.

Configuring identity only and LDAP

Configure IBM® WebSphere® Application Server with the Ant **configure** target to use identity only and LDAP

About this task

To configure WebSphere® Application Server with the Ant **configure** target to use identity only and LDAP, set the *curam.security.check.identity.only* property to true in `AppServer.properties` before running the Ant **configure** target. This step configures the CuramLoginModule behavior to be compatible with the identity only and LDAP authentication mechanism. For a previously configured server see the steps in Set up the system JAAS login module on page 37. to set the corresponding *CuramLoginModule check_identity_only* property.

However, when using identity only in combination with WebSphere® Application Server and LDAP you might need to perform additional manual configuration steps; this is regardless of whether configuration is done using the WebSphere® Application Server Administrative Console or the Ant **configure** target. For more information, see Identity Only Authentication.

Troubleshooting Identity Only and LDAP

With this combination you may find that WebSphere® Application Server fails to start successfully because you must add a WebSphere Application Server for z/OS-generated username to the login module exclude list property (exclude_usernames) described in Set up the system JAAS login module on page 37. If the application server fails to start there will be a SECJ0270E error message in the `SystemOut.log` file before the failure.

These are the steps needed to resolve this error:

1. Identify the username that is causing the application server start to fail. Configure the login module trace as described in Logging the authentication process on page 20 (in regard to the `configure` target) or Add the login module on page 37 (in regard to configuring via the Administrative Console), and restart WebSphere Application Server. With the login module trace running, prior to the SECJ0270E error in the `SystemOut.log` file, the trace data will identify the failing username with a record like this:

```
SystemOut    O Username: server:MyNodeCell_MyNode_CuramServer
```

Where "MyNode" is the node name, "MyNodeCell" is the cell name, and "CuramServer" is the WebSphere server name. Following the login module trace data will be the error, which will look like this:

```
SECJ0270E: Failed to get actual credentials.
   The exception is javax.security.auth.login.LoginException:
   Context: MyNodeCell/nodes/MyNode/servers/CuramServer,
   name: curamejb/LoginHome:
```

```
        First component in name curamejb/LoginHome not found.
```
2. Specify the failing username in the login module exclude_usernames property in the
   WebSphere Application Server configuration. Since WebSphere Application Server is failing
   to start you cannot make this change via the Administrative Console and you must edit the
   WebSphere Application Server configuration file directly. In the WebSphere Application
   Server configuration file system edit *config\cells\MyNodeCell\security.xml*,
   which will have three occurrences of the exclude_usernames property (one for each alias);
   e.g.:

```
<options xmi:id="Property_1301940482165"
   name="exclude_usernames"
   value="websphere,db2admin"
   required="false"/>
```

You must modify the three occurrences to include the newly identified username from the
trace entry above; e.g.:

```
<options xmi:id="Property_1301940482165"
   name="exclude_usernames"

 value="websphere,db2admin,server:MyNodeCell_MyNode_CuramServer"
   required="false"/>
```

Note that in the exclude_usernames occurrences the id attribute will vary per your system
configuration and the comma separator in the example value attribute represents the default
curam.security.usernames.delimiter value, which may be different in your case.
3. Restart WebSphere Application Server.

User registry

When the *CuramLoginModule* login module is configured for identity only, the IBM®
WebSphere® Application Server user registry is queried.

By default the configured user registry is not queried as part of authentication. When
the *CuramLoginModule* login module is configured for identity only, the WebSphere®
Application Server user registry is queried. You can override this default behavior before
running the Ant **configure** target to set the *curam.security.user.registry.enabled* property in
AppServer.properties. For a previously configured server see the steps in Set up the system JAAS
login module on page 37 to set the corresponding *CuramLoginModuleuser_registry_enabled*
property.

If this property is set to true the user registry is queried during the authentication process,
regardless of whether identity only authentication is enabled or disabled. If this property is
set to false, the user registry is not queried. For example, if *curam.security.check.identity.only*
(*check_identity_only*) is set to true and *curam.security.user.registry.enabled*
(*user_registry_enabled*) is set to false, neither the Cúram authentication verifications nor the
WebSphere® Application Server user registry is used as part of the authentication process.

You can also control the authentication of different types of external users against
the user registry by using the *curam.security.user.registry.enabled.types* and/or the
curam.security.user.registry.disabled.types properties in *AppServer.properties* before
running the Ant **configure** target. For a previously configured server see the steps in Set up

the system JAAS login module on page 37 to set the corresponding *CuramLoginModule user_registry_enabled_types* or *user_registry_disabled_types* properties. The following properties specify a comma-delimited list of external user types that might be authenticated using the user registry:

- User types specified in the curam.security.user.registry.enabled.types (user_registry_enabled_types) list are processed against the user registry (for example, LDAP) and your *ExternalAccessSecurity* implementation. See Securing the application for more information on *ExternalAccessSecurity* implementations.
- User types specified in the *curam.security.user.registry.disabled.types* (*user_registry_disabled_types*) list are not processed against the user registry and the processing of your ExternalAccessSecurity implementation is the authority for authentication.

The precedence order in processing these properties and the WebSphere® Application Server user or external (LDAP) registry is as follows:

1. By default the user registry is not checked and the application authentication is used.
2. The setting of the *curam.security.user.registry.enabled* property (*user_registry_enabled*) to true requires authentication by both the WebSphere® Application Server, or external (LDAP), user registry and application security for internal users or your *ExternalAccessSecurity* implementation for external users.
3. An external user of a type specified in the *curam.security.user.registry.enabled.types* (*user_registry_enabled_types*) list must be authenticated by the WebSphere® Application Server, or external, user registry and your *ExternalAccessSecurity* implementation.
4. An external user of a type specified in the *curam.security.user.registry.disabled.types* (*user_registry_disabled_types*) list is not authenticated by the WebSphere® Application Server, or external, user registry and your *ExternalAccessSecurity* implementation is the authority.

Logging the authentication process

Trace entries of the *CuramLoginModule* authentication process can be generated in `SystemOut.log` and these can be helpful for debugging.

To generate trace log entries add the following to `AppServer.properties` before running the Ant **configure** target:

```
curam.security.login.trace=true.
```

For a previously configured server, see the steps in Set up the System JAAS Login Module to set the *CuramLoginModule login_trace* property.

Establishing an alternate exclude username delimiter

In rare cases, usernames might conflict with the character (a comma) used to separate the list of usernames that the *CuramLoginModule* excludes from authentication. In this case an alternate character can be specified.

To specify a different character add for example:

```
curam.security.usernames.delimiter=|
```

to *AppServer.properties* before running the Ant **configure** target.

For a previously configured server, see the steps in [Set up the System JAAS Login Module](#) to set the *exclude_usernames_delimiter* property.

WebSphere® Application Server caching behavior

WebSphere® Application Server caches user information and credentials in a security cache and the login module will not be invoked while a user entry is valid in this cache.

The default invalidation time for this security cache is ten minutes, where the user has been inactive for ten minutes. The *WebSphere Caching Behavior* section in the *Cúram Security Handbook* should be consulted for further details on this.

Security custom properties

Security custom properties determine the behavior of a single sign-on LTPA Token2 login.

- com.ibm.ws.security.webChallengeIfCustomSubjectNotFound

  This property determines the behavior of a single sign-on LTPA Token2 login.

  When this property value is set to `true`, the token contains a custom cache key, and the custom Subject cannot be found, the token is used to log in directly as the custom information needs to be gathered again. A challenge occurs so that the user to login again. When this property value is set to `false` and the custom Subject is not found, the LTPA Token2 is used to login and gather all of the registry attributes. However, the token might not obtain any of the special attributes that downstream applications might expect.

  By default the configuration script sets a WebSphere Application Server property, com.ibm.ws.security.webChallengeIfCustomSubjectNotFound, to `false` to ensure that web sessions can seamlessly transfer between two servers in a cluster (for example, in a fail over scenario) without being asked for security credentials. This setting allows the security token used by WebSphere Application Server to be validated correctly, without user input.

  If this behavior is not required it is possible to change this property to true, see [Set up the system JAAS login module on page 37](#) for more information on setting *Security custom properties*. If the property is set to `true`, when a web session switches from one server in the cluster to another, perhaps due to the original server failing, the user will be asked for security information before being able to proceed.

Security hardening measures

When a user logs into the application, the username and password is sent to the server, and if successfully authenticated, the server responds with a unique 'LTPA token'.

The LTPA token is used in all subsequent requests to recognize the user and then serves privileged content. When the user logs out, this token should become invalid. but this is not the case and there is no way to invalidate the LTPA token, which has been confirmed by IBM. **IBM's recommendation is to use two "security hardening measures" of:**

1. Setting the security Requires SSL option;
2. Setting a custom property to limit LTPA cookies to SSL only.

The default configuration scripts make this change and the steps are documented Configure administration security on page 34.

Cryptography

Use cryptography to encrypt passwords.

Cúram cryptography relates to functionality for managing passwords and is covered in detail in the *Cúram Security Handbook* and you should consult it in consideration that:

- For production environments it is strongly recommended that you modify the default settings.
- For development and test environments you need to consider where the defaults provide acceptable protection in your environment.

1.4 Starting, stopping, and restarting IBM® WebSphere® Application Server

Start, stop, and restart the application server.

A number of Ant targets are provided to start and stop WebSphere® Application Server. Execute the targets from the *<SERVER_DIR>* directory and as for the **configure** target, they require the *AppServer.properties* file to be set up correctly (1.2 Configuring WebSphere® Application Server on page 15). They also require a number of extra parameters to be specified and these are detailed as follows.

Start the application server

The target for starting a server is **startserver** and requires the option -Dserver.name, for example:

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

> **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 the application server

The target for stopping a server is **stopserver** and requires the following option: -Dserver.name, for example:

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

Restart the application server

The target for restarting a server is **restartserver** and the options are the same as for the **startserver** target.

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

1.5 Deploying an application

After packaging the Cúram application and web services application in EAR files, you need to deploy them to the application server.

> **Note:** The configuration scripts provided in IBM® WebSphere® Application Server only support a simple configuration targeted at a single server in the Express or Base editions of WebSphere® Application Server.

SDEJ targets

SDEJ provides targets for installing and un-installing applications on WebSphere® Application Server. As with the **startserver** / **stopserver** targets, the **installapp** / **uninstallapp** targets require that the *AppServer.properties* file is configured correctly (see).

Install an application

You do not have to restart the server after installation, the install target automatically starts the application.

The Ant target to install an application (in the form of an EAR file) is **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 of the application.

An example command is as follows: **build installapp -Dserver.name=CuramServer -Dear.file=$SERVER_DIR/build/ear/Curam.ear -Dapplication.name=Curam**

> **Note:** You must deploy the EAR file containing the server module before installing any other (client-only) EAR files.

You can use the optional Ant property wsadmin.extra.argsto pass additional arguments to wsadmin. The following example sets new Java® heap sizes and passes the option to append wsadmin tracing: **-Dwsadmin.extra.args="-javaoption -Xms1024m -javaoption -Xmx1024m -appendtrace true"**

Do not use this property to set arguments already passed using the Ant scripts. Also, you can observe these scripts when running Ant by specifying its verbose option: -v.

Change SYSTEM Username

You must change the username for JMS invocation while deploying the application by setting the following properties in *AppServer.properties* file before deployment:

- curam.security.credentials.async.username The username that JMS invocations runs under.
- curam.security.credentials.async.password The encrypted password associated with the username. The password must be encrypted using the Ant **encrypt** target. See the *Cúram Server Developers Guide* for more information.

You can also change the username when the application has been deployed by using the WebSphere® Application Server administrative console. Navigate to **Applications** > **Application Types** > **WebSphere enterprise applications** and select the Cúram application. Select the **User RunAs roles** link. Check the everyone role, enter a new username, unencrypted password and click **Apply**. Save the changes as detailed in .

> **Note:** The new username must be 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.

Uninstall an Application

Use the Ant target **uninstall** to uninstall an application. **uninstall** 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.

An example command is as follows: **build uninstallapp -Dserver.name=CuramServer -Dapplication.name=Curam**

1.6 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 JSP s 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/WAS/
Curam.ear
```

Figure 1: Example of Usage

> **Note:** While running the Ant **precompilejsp** target for IBM® WebSphere® Application Server, an out of memory exception may occur (or some JSPs may silently be ignored and not pre-compiled). To work around this, modify the *JspBatchCompiler.bat* script in the *%WAS_HOME%\bin* directory to increase the maximum memory size. Change the memory consumption from -Xmx256m to at least -Xmx1024m.

1.7 Testing the deployment

Start and test the deployed application.

When the application is installed on a configured WebSphere Application Server installation the next step is to start and test the application.

> **Note:** The installation of a web services application might also be required.

Ensure the relevant server is started 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 WebSphere Application Server is running, *<port>* identifies the server port on which client application is deployed and *<context-root>* identifies the Context Root of the WAR module (see Multiple EAR files on page 13, for details). There is no need to restart the server after an application is deployed.

Before the page can be opened, the browser will be directed to the login page. Log in with a valid Cúram 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.

1.8 Manual WebSphere® Application Server configuration

The IBM® WebSphere® Application Server can be configured manually if required, but this is not recommended. This section is for information purposes only.

Any settings entered under the **Resources** section of the IBM® WebSphere® Application Server Administrative Console can be configured at multiple levels that control the JNDI scope. These include cell, node, or server. Upon selecting a **Resource**, the top of the main browser window shows this scope and allows the various resources in the current scope to be viewed. The scope, and in turn the location of any resources set, should be based upon planned use, that is, if working in a cluster it may not be necessary to set the same settings on each server, so the scope may be set to cell or node.

The administrative console

Configure the IBM® WebSphere® Application Server by using the Administrative Console. To run the administrative console, start the default server, e.g.server1, because theconsole is installed as a web application on this server.

To start server1, use the *startServer.bat*, located in the *profiles/AppSvr01/bin* directory of the IBM® WebSphere® Application Server as follows:

<WEBSPHERE INSTALL DIR>/profiles/AppSvr01/bin/startServer server1

Point a web browser at the following URL:

```
http://localhost:9060/ibm/console"/>
```

Alternatively, start the Administrative Console from **Start** > **Programs** > **IBM WebSphere** > **Application Server V8.5** > **Profiles** > **AppSvr01** > **Administrative console**. The **Start the server** and **Stop the server** commands can also be used from this menu to start and stop the servers.

The first time the Administration Console is opened, a username is requested to log in. This username can be anything. The Administration Console is divided into two sections. The left hand side contains a tree hierarchy for navigating the console and the right hand side displays the information related to the current node selected in the tree. When instructed toNavigate to, the tree hierarchy should be traversed to the relevant node.

Scripting support

Change the IBM® WebSphere® Application Server property files to support the execution of provided Ant scripts.

sas.client.props

Open *sas.client.props* in the *profiles/AppSvr01/properties* directory. Set the login source to retrieve the username and password from a properties file. Set, or where necessary, add the following properties:

```
com.ibm.CORBA.loginSource=properties
# RMI/IIOP user identity
com.ibm.CORBA.loginUserid=websphere
com.ibm.CORBA.loginPassword=websphere
```

Where *websphere* is the username and password for the Administration Console.

soap.client.props

Open *soap.client.props* in *profiles/AppSvr01/properties*. Set the login source to retrieve the username and password from a properties file. Set the following properties to match the credentials you configured for WebSphere® Application Server as in 1.2 Configuring WebSphere® Application Server on page 15. In the following sample the values are examples and the password specified in this file cannot be encrypted:

```
com.ibm.SOAP.loginUserid=websphere
com.ibm.SOAP.loginPassword=websphere
```

where *websphere* is the username and password for the Administrative Console.

To avoid timeouts when installing EAR files ensure that the following is at least:

```
com.ibm.SOAP.requestTimeout=3600
```

server.policy

Open *server.policy* in the *profiles/AppSvr01/properties* directory. Add the following lines to the end of this file:

```
grant codeBase "file:<CURAMSDEJ>/drivers/-" {
permission java.security.AllPermission;
};
```

where *<CURAMSDEJ>* is the SDEJ installation directory.

```
grant codeBase "file:${was.install.root}/
profiles/<profile.name>/installedApps/
<cell.name>/<SERVER_MODEL_NAME>.ear/
guice-2.0.jar" { permission java.lang.RuntimePermission
"modifyThread"; permission java.lang.RuntimePermission
"modifyThreadGroup"; };
```

Where:

1. *<profile.name>* is the name of the target WebSphere® Application Server profile.
2. *<cell.name>* is the name of the target WebSphere Application Server cell.
3. *<SERVER_MODEL_NAME>* is the name of the application (base name of the EAR file).

Creating the data source login alias

Create the data source login alias. IBM® DB2® , IBM DB2 for z/OS® , and Oracle® Database are the databases supported.

About this task

Use the administrative console to configure a login alias for both the DB2 and Oracle data sources as follows:

Procedure

1. Navigate to **Security** > **Global security**;
2. Expand the **Java Authentication and Authorization Service** option in the **Authentication** section and select the **J2C authentication data** option;
3. Click **New** to open the Configuration screen;
4. Set the following fields:

   Alias = dbadmin

   User ID = *<database username>*

   Password = *<database password>*

   Description = The database security alias

   where *<database username>* and *<database password>* are set to the username and password used to login to the database;
5. Click **OK** to confirm the changes.

Configure Db2® data sources

Configure the Db2® environment variable, the database driver provider, and the database driver data source.

Configure the Db2® environment variable
Select the `DB2_JCC_DRIVER_PATH` link and configure the environment variable.

Procedure

1. Select **Environment** > **WebSphere variables**.
2. Select the `DB2_JCC_DRIVER_PATH` link from the list of environment variables. A configuration screen for the variable is displayed.

3. Set the **Value** field so that the field points to the directory that contains the Type 4 drivers. Typically, the directory is the `drivers` directory that is under the SDEJ installation. For example, `D:\Curam\CuramSDEJ\drivers`.

4. Select **OK** to confirm the changes.

Configure the database driver provider

Use the **Resources** menu to add the driver and configure the properties.

Procedure

1. Select **Resources** > **JDBC** > **JDBC providers**.

2. Ensure that the appropriate scope to define the data source is selected.

3. Select **New** to add a driver. A configuration screen is displayed.

4. Select **DB2** from the **database type** drop-down list.

5. Select **DB2 Using IBM JCC Driver** from the **Provider type** drop-down list.

6. Select **XA data source** from the **Implementation type** drop-down list.

7. Select **Next** to continue.

8. Review the configuration properties. Typically, no change is required unless you want to connect to a IBM® z/OS® database. To connect to an IBM® z/OS® database, verify that the `${DB2_JCC_DRIVER_PATH}` field points to the correct directory for your system. For example, ensure that the field points to the directory that contains the IBM® Db2® Connect license jar `db2jcc_license_cisuz.jar` that is provided by IBM® z/OS® connectivity.

9. Select **Next**.

10. Select **Finish** to confirm the changes.

Configure the database driver data source

Add the data source for the database driver and configure the properties for the data source.

About this task

Perform the following list of steps for each of the data sources. For each data source, substitute `curamdb`, `curamsibdb`, and `curamtimerdb` for *<DatasourceName>*.

> **Note:** In the data source name, do not use angle brackets.

Procedure

1. Select **DB2 Using IBM JCC Driver(XA)** that is displayed on the **JDBC Providers** list. The configuration screen for the provider is displayed.

2. Select the **Data sources** link that is listed under **Additional Properties**.

3. Select **New** to add a data source.

4. Use the following list of properties for the fields:

   - **Data source name** *<DatasourceName>*
   - **JNDI name** *jdbc/<DatasourceName>*

5. Select **Next** to continue.

6. Use the following list of properties for the fields:

   - **Driver type** 4.
   - **Database name** The name of the Db2 database.
   - **Server name** The name of the Db2 database server.
   - **Port number** The Db2 database server port.

   Ensure that the **Use this data source in container managed persistence (CMP)** checkbox is selected.

7. Select **Next**.

8. Use the following list of properties for the fields:

   - Set the **Component-managed authentication alias** drop-down value to *<valid for database>*.
   - Set the **Mapping-configuration alias** drop-down value to DefaultPrinicipalMapping.
   - Set the **Container-managed authentication alias** drop-down value to *<valid for database>* where the *<valid for database>* alias used is the alias that you configured in the security setting. For more information, see the *Creating the Data Source Login Alias* related link.

   Unless a specific change is required, do not modify the other fields.

9. Select **Next** to continue.

10. Select **Finish** to confirm the changes and continue.

11. Select the newly created *DatasourceName* data source from the displayed list.

12. Select the **Custom Properties** link in **Additional Properties**.

13. Select fullyMaterializeLobData.

14. Set the value to **false**.

15. Select **OK** to confirm the change.

Related tasks

Creating the data source login alias on page 28
Create the data source login alias. IBM® DB2® , IBM DB2 for z/OS® , and Oracle® Database are the databases supported.

Configure an Oracle Database data source

Configure an Oracle Database and database drivers.

Configure the Oracle environment variable
Use the **Environment** menu to select and configure the Oracle environment variable.

Procedure

1. Navigate to **Environment** > **WebSphere variables**.

2. Select the ORACLE_JDBC_DRIVER_PATH link from the list of environment variables. This will open the configuration screen for this variable.

3. Set the **Value** field to point to the directory containing the Type 4 driver. This is the *drivers* directory of the SDEJ installation, e.g. *D:\Curam\CuramSDEJ\drivers.*

4. Select **OK** to confirm the changes.

Configure the XA database driver
Add an XA database driver and ensure that the configuration properties are correct.

Procedure

1. Select **Resources** > **JDBC** > **JDBC providers**.
2. Ensure that the appropriate scope for where to define the data source is selected.
3. Select **New** to add a driver. A configuration screen is displayed.
4. Select **Oracle** from the **Database type** drop-down list.
5. Select **Oracle JDBC Driver** from the **Provider type** drop-down list.
6. Select **XA data source** from the **Implementation type** drop-down list.
7. Set the **Name** field to **Oracle JDBC Driver (XA)**, where the field is not automatically completed.
8. Select **Next** to continue.
9. Review the **Class path** and ensure that the ORACLE_JDBC_DRIVER_PATH environment variable is correct.
10. Select **Next** to continue. A configuration screen is displayed.
11. Review the configuration properties. Typically, changing the properties is not required.
12. Select **Finish** to confirm the changes.

Configure the non-XA database driver
Use the **Resources** menu to add and configure the non-XA database driver.

Procedure

1. Navigate to **Resources** > **JDBC** > **JDBC providers**.
2. Select **New** to add a new driver. This will open a configuration screen.
3. Select **Oracle** from the the list in the **Database type** drop down supplied.
4. Select **Oracle JDBC Driver** from the list in the **Provider type** drop down supplied.
5. Select the **Connection pool data source** from the list in the **Implementation type** drop down supplied.
6. Set the **Name** field to be Oracle JDBC Driver, if not filled in automatically.
7. Select **Next** to continue.
8. Review the **Class path** and ensure the ORACLE_JDBC_DRIVER_PATH environment variable is correct. Select **Next** to continue.
9. Review the properties on the configuration screen that opens. There should be no need to change any of them.
10. Select **Finish** to confirm the changes.

Configure the XA database driver data sources

Add a data source and configure the properties.

About this task

The following steps should be repeated twice, substituting *<DatasourceName>* (without the angle brackets) with the curamdb and thencuramsibdb.

Procedure

1. Select the **DB2 Using IBM JCC Driver(XA)** displayed on the list of existing providers. This opens the configuration screen again.

2. Select the **Data sources** link under **Additional Properties**.

3. Select **New** to add a new data source.

4. Set the fields as follows:

    -   **Data source name** : *<DatasourceName>*
    -   **JNDI Name** : *jdbc/<DatasourceName>*

5. Select **Next**.

6. Set the **URL** Value field to:

    `jdbc:oracle:thin:@//serverName:port/databaseServiceName`, to connect to database using Oracle service name.

    or

    `jdbc:oracle:thin:@serverName:port:databaseName`, to connect to database using Oracle SID name.

    Where:

    -   *serverName* is the name of the server hosting the database.
    -   *port* is the port number the database is listening on.
    -   *databaseName* is the SID of the database.
    -   *databaseServiceName* is the service name of the database.

7. Set the **Data store helper class name** to be `Oracle 11g data store helper`.

8. Leave all other fields untouched unless a specific change is required and select **Next**.

    > **Note:** Oracle recommends using the **URL** format `jdbc:oracle:thin:/@//serverName:port/databaseServiceName` to connect to Oracle database using service name. But this **URL** format(extra '/' before the '@' in the **URL**) is not supported by the WebSphere Application Server admin console. So, the Oracle service name **URL** described above (without extra '/' before the '@' in the URL) should be used while configuring Oracle data source from admin console, to connect to Oracle database using service name.

9. Set the **Authentication alias for XA recovery** to: *<valid for database>*

    where the *<valid for database>* alias used is the one set up in Creating the data source login alias on page 28.

Leave all other fields unchanged unless a specific change is required and select **Next**.

10. Set the **Component-managed authentication alias** drop down value to: *<valid for database>*.

11. Select **Finish** to confirm the changes and continue.

Configure the non-XA database driver data source

Add the data source for the database driver and configure the properties for the data source.

Procedure

1. Select the Oracle JDBC Driver displayed on the list of existing providers. This opens the configuration screen again.

2. Select the **Data sources** link under **Additional Properties**.

3. Select **New** to add a new data source.

4. Set the fields as follows:

   - **Data source name** :curamtimerdb
   - **JNDI Name** :jdbc/curamtimerdb

5. Select **Next**.

6. Set the **URL** Value field to:

   jdbc:oracle:thin:@//serverName:port/databaseServiceName, to connect to database using Oracle service name.

   or

   jdbc:oracle:thin:@serverName:port:databaseName, to connect to database using Oracle SID name.

   Where:

   - *serverName* is the name of the server hosting the database.
   - *port* is the port number the database is listening on.
   - *databaseName* is the SID of the database.
   - *databaseServiceName* is the service name of the database.

7. Set the **Data store helper class name** to be `Oracle 11g data store helper`.

8. Leave all other fields untouched unless a specific change is required and select **Next**.

> **Note:** Oracle recommends to use the **URL** format *jdbc:oracle:thin:/@// serverName:port/databaseServiceName* to connect to Oracle database using service name. But this **URL** format(extra '/' before the '@' in the **URL**) is not supported by WebSphere administrative console. So, the Oracle service name **URL** described above (without extra '/' before the '@' in the URL) should be used while configuring Oracle data source from admin console, to connect to Oracle database using service name.

9. Set the **Component-managed authentication alias** drop down value to: *<valid for database>*.

where the *<valid for database>* alias used is the one set up in .

10. Leave all other fields unchanged unless a specific change is required and select **Next**.

11. Select **Finish** to confirm the changes and continue.

Save the master configuration

Save the master configuration

Select the **Save** link in the **Message(s)** box. This box is displayed only after configuration changes have been made.

Configure administration security

Configure administration security on the IBM® WebSphere® Application Server administration interface.

About this task

The default user registry used is the default WebSphere® Application Server file- based user registry.

Procedure

1. Navigate to **Security** > **Global security**.

2. Set the **Available realm definitions** to be **Federated repositories** and click the **Configure** button.

3. Set the **Primary administrative username** to be `websphere`.

4. Select the **Automatically generated server identity** radio button.

5. Select **Ignore case for authorization** and click **OK**.

6. Enter the password for the default administrative user, e.g. `websphere`, enter the confirmation and click **OK** to confirm the changes.

7. Set the **Available realm definitions** to be **Federated repositories** and click the **Set as current** button.

8. Select **Enable administrative security**.

9. Select **Enable application security**.

10. Clear the **Use Java 2 security to restrict application access to local resources** option.

11. Click the **Apply** button to confirm the changes.

12. Set the **Available realm definitions** to be **Federated repositories**

13. Click the **Apply** button to confirm the changes.

14. Navigate to **Security** > **Global security**.

15. Expand **Web and SIP Security** and select **Single sign-on (SSO)**.

16. Select **Requires SSL**.

17. Click **OK** to confirm the change

18. Navigate to **Security** > **Global Security**

19. Select the **Custom Properties** link.

20. Click the **New** button and set the name and value as follows:

   Name : `com.ibm.ws.security.web.logoutOnHTTPSessionExpire`

   Value : `true`

21. Click the **OK** button to add the new property.

22. Click the **New** button and set the name and value as follows:

   Name : `com.ibm.ws.security.addHttpOnlyAttributeToCookies`

   Value : `true`

23. Click **OK** to confirm the change

24. Save the changes to the master configuration.

Configure transport layer security

Set the Transport Layer Security (TLS) protocol to TLS v1.2.

About this task

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

Procedure

1. Navigate to **Security** > **SSL certificate and key management**.

2. Select **SSL configurations**.

3. Select a specific configuration, e.g. NodeDefaultSSLSettings and open it. This process needs to be carried out for all of the configurations listed.

4. Under the Additional Properties, click **Quality of Protection (QoP) settings**.

5. From the **Protocol** drop-down, Select **TLSv1.2**.

6. For the **Cipher suite settings** drop-down, ensure that **Strong** is selected and then click **Update selected ciphers**.

   > **Note:** 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.

7. Select **Apply** to confirm the changes.

8. Select **OK** to confirm the change.

9. Save the changes to the master configuration.

10. Edit the **ssl.client.props** file located in the folder **<WAS_HOME>/profiles/<Profile Name>/properties**. Find the property `com.ibm.ssl.protocol` and set its value to `TLSv1.2` e.g. `com.ibm.ssl.protoco=TLSv1.2`.

11. Save the file.

Restart the application server

Restart the servers that the security changes take effect and to add additional required users.

This step is compulsory. Stop the server by using the `stopServer.bat` file in the `profiles/AppSrv01/bin` directory of the IBM® WebSphere® Application Server installation.

Before restarting the application server (e.g. server1), you must make the registry and cryptography JAR files available to WebSphere® Application Server. The registry JAR file contains classes necessary for the security configuration and the cryptography JAR file contains necessary configuration settings and data for password security.

The `Registry.jar` file is located in the `lib` directory of the SDEJ installation. Copy this file into the `lib` directory of the WebSphere® Application Server installation.

Generate the `CryptoConfig.jar` file by running the ant target **configtest** as follows, *build configtest -Dcrypto.ext.dir=filedir*. Copy the `CryptoConfig.jar` from the generated location. Copy this file into the Java® `jre/lib/ext` directory. If you require customizations to the cryptographic configuration see the *Curam Security Handbook* for more information.

Start the application server (e.g. server1) and open the Administrative Console to continue with the configuration steps. Because the security configuration is complete and the scripting changes have been made, you can now use the SDEJ scripts to restart the WebSphere® Application Server. See for more details on restarting the server.

Open the Administrative Console to continue with the configuration. Now that global security is enabled, you must log into the console with the username and password credentials set up previously.

Add users to the user registry

Add the IBM® WebSphere® Application Server administrative users and the database user must be manually added to the user registry.

About this task

As detailed in , the configured WebSphere® Application Server user registry is used for authentication of administrative users and the database user. The WebSphere administrative users and the database user must be manually added to the user registry as follows.

Procedure

1. Navigate to **Users and Groups** > **Manage Users**.
2. Select **Create**.
3. Fill in the details for the WebSphere® Application Server administrative user and select **Create**.
4. Repeat the steps for the database user.

Results

If WebSphere® Application Server administrative security was enabled when creating the profile the administrative user may already be defined in the registry.

Set up the system JAAS login module

Application security uses a JAAS (Java Authentication and Authorization Service) login module for authentication. This login module must be configured for the *DEFAULT*, *WEB_INBOUND* and *RMI_INBOUND* configurations. Repeat the steps for each of these configurations.

Add the login module
Add the login module.

Procedure

1. Navigate to **Security** > **Global security**.

2. Expand **Java Authentication and Authorization Service** entry in the **Authentication** section and select **System logins**.

3. Select the relevant Alias from the list. Configure the login module for the *DEFAULT*, *WEB_INBOUND* and *RMI_INBOUND* aliases.

4. Select **New** to configure a new Login Module.

5. Set the **Module class name** field to be `curam.util.security.CuramLoginModule`.

6. Check the **Use login module proxy** option.

7. Select **REQUIRED** in the **Authentication strategy** field.

8. Enter into **Custom properties** table Name/Value pairs for any required properties as listed in Table 1, select **New** as needed.

Table 1: CuramLoginModule Custom Properties

Name	Example Value	Description
exclude_usernames	websphere, db2admin	Required. A list of usernames to be excluded from authentication. The default delimiter is a comma, but may be overridden by exclude_usernames_delimiter. This list should include the WebSphere administration user (as specified in Configure administration security on page 34) and the database user (as specified in Creating the data source login alias on page 28). Any users listed here should be defined in the WebSphere Application Server user registry.
exclude_usernames_delimiter	|	*Optional.* A delimiter for the list of usernames provided in exclude_usernames. A delimiter other than the default comma can be useful when usernames have embedded commas as with LDAP users.
login_trace	true	*Optional.* This property should be set to true to debug the authentication process. If set to true the invocation of the login module will result in tracing information being added to the WebSphere Application Server *SystemOut.log* file.

Name	Example Value	Description
module_name	DEFAULT, WEB_INBOUND or RMI_INBOUND	*Optional.* This property should be set to one of DEFAULT, WEB_INBOUND or RMI_INBOUND depending on the configuration the login module is being defined for. It is used only when login_trace is set to true for tracing purposes.
check_identity_only	true	*Optional.* If this property is set to true the login module will not perform the usual authentication verifications. Instead it will simply ensure that the user exists on the database table. In this case the configured WebSphere Application Server user registry will not be by-passed and will be queried after the login module. This option is intended where LDAP support is required or an alternative authentication mechanism is to be used. **Note:** If you are specifying identity only and using LDAP you may need to perform additional configuration steps; please see Configuring identity only and LDAP on page 18.
user_registry_enabled	true	*Optional.* This property is used to override the behavior of by-passing the user registry. If this property is set to true the WebSphere Application Server user registry will be queried during the authentication process. If this property is set to false, the WebSphere Application Server user registry will not be queried.
user_registry_enabled_types	EXTERNAL	*Optional.* This property is used to specify a comma-delimited list of external user types that will be processed against the WebSphere Application Server user registry (e.g. LDAP). See User registry on page 19 for more information on the processing of the WebSphere Application Server user registry.
user_registry_disabled_types	EXTGEN,EXTAUTO	*Optional.* This property is used to specify a comma-delimited list of external user types that will not be processed against the WebSphere Application Server user registry (e.g. LDAP). See User registry on page 19 for more information on the processing of the WebSphere Application Server user registry.

9. Select **OK** to confirm the addition of the new login module.

Reorder the login module

Reorder the login module for the *DEFAULT*, *WEB_INBOUND*, and *RMI_INBOUND* aliases.

Procedure

1. Navigate to **Security** > **Global security**.
2. Expand **Java Authentication and Authorization Service** in the **Authentication** section and select **System logins**.
3. Select the relevant Alias from the list. Reorder the login module for the *DEFAULT*, *WEB_INBOUND*, and *RMI_INBOUND* aliases as follows:
   a) Select **Set Order**.
   b) Select **curam.util.security.CuramLoginModule** and select **Move Up**. Repeat this until the CuramLoginModule entry is the top entry in the list
4. Select **OK** to confirm the modifications to the order.

Disable cross-cluster authentication

Disable cross-cluster authentication.

About this task

This property *com.ibm.ws.security.webChallengeIfCustomSubjectNotFound* determines the behavior of a single sign-on LTPA Token2 login. The property is set to false to ensure that web sessions can seamlessly transfer between two servers in a cluster (for example, in a failover scenario) without being asked for security credentials.

Procedure

1. Navigate to **Security** > **Global security**.
2. Select **Custom properties** and select the **com.ibm.ws.security.webChallengeIfCustomSubjectNotFound** property from the list of available properties.
3. Under **General Properties**, change the value of the **com.ibm.ws.security.webChallengeIfCustomSubjectNotFound** property to *false*
4. Select **OK** to confirm the addition.
5. Save the changes to the master configuration as described in <u>Save the master configuration on page 34</u>.

Configuring the server

Configure server settings such as JNDI lookup port and Java Virtual Machine.

Configure the JNDI lookup port

Configure the JNDI lookup port

Procedure

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.
2. Select the relevant server from the list, e.g. server1.
3. Expand **Ports** in the **Communications** section and select **Details**.
4. Select the **BOOTSTRAP_ADDRESS** entry and set the **Port** to match the value of the property curam.server.port in your `AppServer.properties` file.
5. Select **OK** to apply changes.
6. **Save** the changes made to the master configuration.

Configure the ORB pass by reference

Configure the ORB pass by reference

Procedure

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.
2. Select the relevant server from the list, e.g. server1.

3. In the **Container Settings** section expand **Container Services** and Select the **ORB service** link.

4. Select the **Pass by reference** option from the **General Properties** section.

5. Click **OK** to apply changes.

6. **Save** the changes made to the master configuration.

Configure the Java Virtual Machine

Configure the Java Virtual Machine (JVM).

Procedure

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.

2. Select the appropriate server from the list.

3. In the **Server Infrastructure** section expand **Java and Process Management**.

4. Select the **Process definition** link.

5. In the **Additional Properties** section Select the **Java Virtual Machine** link.

6. Set the fields as follows:

   Initial heap size: 1024

   Maximum heap size: 1024

   Select **Apply** to set the values.

7. In the **Additional Properties** section, select the **Custom Properties** link.

8. Select **New** and set the properties as follows:

   - **Name**:
     `com.ibm.websphere.security.util.authCacheCustomKeySupport`
   - **Value**: `false`
   - Select **OK** to add the property

9. *This step is only required on non-Windows platforms.*

   Select **New** and set the properties as follows:

   Name : `java.awt.headless`

   Value : `true`

   Select **OK** to add the property

10. **Save** the changes made to the master configuration.

Configure the timer service

Configure the timer service

Procedure

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.

2. Select the appropriate server from the list.

3. In the **Container Settings** section expand **EJB Container Settings**.

4. Select the **EJB timer service settings** link;

5. In the **Scheduler Type** panel select the **Use internal EJB timer service scheduler instance** option;

6. Set the fields as follows:

   - **Data source JNDI name**: jdbc/curamtimerdb
   - **Data source alias** : *<valid for database>*

     where the alias used is the one set up in <u>Creating the data source login alias on page 28</u>.

7. Select **OK** to confirm the changes.

8. **Save** the changes made to the master configuration.

Configure the port access

Configure the port access.

Procedure

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.

2. Select the appropriate server from the list.

3. Select the **Ports** link in the **Communications** section.

4. Click **details**.

5. Click **New** and set the following fields for the Client TCP/IP port:

   - **User-defined Port Name** : `CuramClientEndPoint`
   - **Host**: *
   - **Port**: `9044`

6. Select **OK** to apply the changes.

7. Select **New** and set the following fields for the WebServices TCP/IP port:

   - **User-defined Port Name**: `CuramWebServicesEndPoint`
   - **Host**: *
   - **Port**: `9082`

8. Select **OK** to apply the changes.

9. Navigate to **Servers** > **Server Types** > **WebSphere application Servers**.

10. Select the relevant server from the list.

11. Expand the **Web Container Settings** branch in the **Container Settings** section.

12. Select the **Web container transport chains** link;

13. Select **New** and set the following fields for the Client transport chain:

    1. **Name**: `CuramClientChain`
    2. **Transport Chain Template**: WebContainer-Secure
    3. Select **Next**
    4. **Use Existing Port**: CuramClientEndPoint
    5. Select **Next** > **Finish**

14. Select **New** and set the following fields for the WebServices transport chain:

1. **Name**: `CuramWebServicesChain`
2. **Transport Chain Template**: WebContainer
3. Select **Next**
4. **Use Existing Port**: CuramWebServicesEndPoint
5. Select **Next** > **Finish**

15. Select the newly created **CuramClientChain**.

16. Select the **HTTP Inbound Channel** link.

17. Ensure the **Use persistent keep alive connections** check box is checked.

18. Select **OK** to confirm the addition.

19. Navigate to **Environment** > **Virtual hosts**.

20. Select **New** to add a new `Virtual Host` by setting the following fields

    Name = *client_host*

    Repeat this step, replacing *client_host* with *webservices_host*.

21. Select the **client_host** link from the list of virtual hosts.

22. Select the **Host Aliases** link in the **Additional Properties** section.

23. Select **New** to add a new `Alias` by setting the following fields.

    - **Host Name** = *
    - **Port** = *9044*

        where *9044* is the port used in step 5. Repeat this step for the other Virtual Host and port used (e.g. webservices_host, 9082);

24. Select **OK** to confirm the addition;

25. Save the changes to the master configuration as described in .

Configure session security integration

Configure session security integration. This setting is required for Cúram web applications.

Procedure

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.

2. Select the relevant server from the list.

3. Select **Session management** in the **Container Settings** section

4. Un-check **Security integration**.

5. Select **OK** to apply changes.

6. **Save** the changes made to the master configuration.

Configure the Service Integration Bus

Configure the Service Integration Bus

Procedure

1.   Navigate to **Service integration** > **Buses**.

2.   Select **New** and in **Step 1** set the following field:

   Name: CuramBus

   Leave everything else as the default and select **Next**.

3.   Entering the **Configure bus security** Wizard, Step 1.1, select **Next**.

   a) In **Step 1.2** of the **Configure bus security** Wizard take the default setting and select **Next**.

   b) In **Step 1.3** of the **Configure bus security** Wizard take the default setting, as appropriate, and select **Next**.

   c) In **Step 1.4** of the **Configure bus security** Wizard review your settings and select **Next**.

4.   In Step 2, select **Finish** to apply the changes.

5.   Select the **CuramBus** now displayed on the list of Buses. This opens the configuration screen.

6.   Select **Bus members** in the **Topology** section.

7.   Select **Add** to open the **Add a New Bus Member** wizard.

8.   Select the server to add to the Bus and select **Next**.

9.   Select **Data store** and select **Next**.

10. Select the option to **use existing data source** and set the options as follows:

   • **Data source JNDI name** = jdbc/curamsibdb
   • **Schema name** = *username*, where *username* is the database username.
   • Deselect the **Create tables** option.
   • Leave everything else as the default and select **Next**.

11. Take the default tuning parameters as appropriate and select **Next**.

12. Select **Finish** to complete and exit the Wizard.

13. Navigate to **Service integration** > **Buses**.

14. Select the **CuramBus** now displayed on the list of Buses. This will open the configuration screen.

15. Select **Security** in the **Additional Properties** section.

16. Select **Users and groups in the bus connector role** in the **Authorization Policy** section.

17. Select **New** to open the **SIB Security Resource Wizard**.

18. Select the **The built in special groups** radio button and select **Next**.

19. Select the **Server** and **AllAuthenticated** check boxes and select **Next**.

20. Select **Finish** to complete and exit the Wizard.

21. Save the changes to the master configuration as described in .

Configuring JMS

Configure JMS components for example, queues and topics.

Configure the JMS connection factories

Configure the JMS connection factories

Procedure

1. Navigate to **Resources** > **JMS** > **JMS providers**.
2. *Note:* The appropriate scope where the JMS resources are to be defined should be selected.
3. Select the **Default messaging provider** link.
4. Select the **Connection factories** link in the **Additional Properties** section.
5. Click **New** and set the following fields:

   - **Name**: CuramQueueConnectionFactory
   - **JNDI Name**: jms/CuramQueueConnectionFactory
   - **Description**: The factory for all connections to application queues
   - **Bus Name**: CuramBus
   - **Authentication alias for XA recovery**: Same as for the `jdbc/curamdb` data source, for example *<SERVERNAME>* /db admin
   - **Mapping-configuration alias**: DefaultPrinicipalMapping
   - **Container-managed authentication alias**: Same as for the Authentication alias for XA recovery.

   Leave everything else as the default and select **OK** to apply the changes.
6. Select **New** and set the following fields:

   - **Name**: CuramTopicConnectionFactory
   - **JNDI Name**: jms/CuramTopicConnectionFactory
   - **Description**: The factory for all connections to application queues.
   - **Bus Name**: CuramBus
   - **Authentication alias for XA recovery**: Same as for the `jdbc/curamdb` data source, for example *<SERVERNAME>* /db admin.
   - **Mapping-configuration alias**: DefaultPrinicipalMapping
   - **Container-managed authentication alias**: Same as for the `jdbc/curamdb` data source, for example *<SERVERNAME>* /dbadmin

   Leave everything else as the default and select **OK** to apply the changes;
7. Save the changes to the master configuration as described in .

Results

> **Note:** With the manual configuration steps it is not possible to correctly configure security for the queue and topic connection factories. To complete this part of the configuration you must use the `wsadmin` tool. To do so exit the Administrative Console and follow these steps:
>
> 1. Identify the queue and topic connection factory entries in the WebSphere Application Server configuration *resources.xml* file. This file resides in the *%WAS_HOME %\profiles\<profile_name>\config* file system hierarchy depending on your naming conventions and the scope where you defined your JMS resources. For instance, using a node-level scope with a profile name of `AppSrv01`, a cell name of `MyNodeCell` and a node name of `MyNode` you would find this file here: *C: \WebSphere\profiles\AppSrv01\config\cells\MyNodeCell\nodes \MyNode\resources.xml*. In this file you must find the `<factories>` entities for the `CuramQueueConnectionFactory` and `CuramTopicConnectionFactory` and make note of the ID for each that begins `J2CConnectionFactory_` followed by a numeric (e.g. 1264085551611).
> 2. Invoke the `wsadmin` WebSphere script. In these examples the language is JACL, so the -*lang jacl* argument may need to be specified along with login credentials, etc. depending on your local configuration.
> 3. In `wsadmin` invoke the following commands; again, assuming node-scope definitions, a cell name of `MyNodeCell`, and a node name of `MyNode`, the resource IDs will be different in your environment.
>
>   1. Get the node and cell identifier: `$AdminConfig getid /Node:MyNode`
>   2. Using the node and cell identifier from the previous step, combine it and the connection factory identifier you obtained above to display the connection factory: `$AdminTask showSIBJMSConnectionFactory CuramQueueConnectionFactory(cells/MyNodeCell/nodes/MyNode| resources.xml#J2CConnectionFactory_1264085551611)`
>
>   From the above command output you should verify that authDataAlias is not set (e.g. `authDataAlias=`), else you're done, as shown in this sample wsadmin output:
>
>       ```
>       {password=, logMissingTransactionContext=false,
>       readAhead=Default, providerEndpoints=,
>       shareDurableSubscriptions=InCluster,
>       targetTransportChain=, authDataAlias=, userName=,
>       targetSignificance=Preferred,
>       shareDataSourceWithCMP=false,
>       nonPersistentMapping=ExpressNonPersistent,
>       persistentMapping=ReliablePersistent, clientID=,
>       jndiName=jms/CuramQueueConnectionFactory,
>       manageCachedHandles=false,
>       consumerDoesNotModifyPayloadAfterGet=false,
>       category=, targetType=BusMember, busName=CuramBus,
>       description=None,
>       xaRecoveryAuthAlias=crouch/databaseAlias,
>       temporaryTopicNamePrefix=, remoteProtocol=,
>       producerDoesNotModifyPayloadAfterSet=false,
>       connectionProximity=Bus, target=,
>       temporaryQueueNamePrefix=,
>       name=CuramQueueConnectionFactory}
>       ```
>   3. To set the authDataAlias use the same connection factory information as above; e.g.: `$AdminTask modifySIBJMSConnectionFactory CuramQueueConnectionFactory(cells/MyNodeCell/nodes/MyNode| resources.xml#J2CConnectionFactory_1264085551611) {-`

Configure the required queues

Configure the required queues.

About this task

Perform the following steps, substituting *<QueueName>* (without the angle brackets) with each of the following queue names: **DPEnactment**, **DPError**, **CuramDeadMessageQueue**, **WorkflowActivity**, **WorkflowEnactment** and **WorkflowError**.

Procedure

1. Navigate to **Service integration** > **Buses** > **CuramBus**.
2. Select the **Destinations** link in the **Destination resources** section.
3. Select **New** to open the **Create new destination** wizard.
4. Select **Queue** as the destination type and click **Next**;
5. Set the following queue attributes:

    Identifier: SIB_ *<QueueName>*

    Leave everything else as the default and select **Next**.
6. Use the **Selected Bus Member** and select **Next**.
7. Select **Finish** to confirm the queue creation.
8. Select the newly added SIB_ *<QueueName>* queue now displayed on the list of existing providers. This action opens the configuration screen again;
9. Use table 1 to set the Exception Destination using the **Specify** radio button and associated text:

    Table 2: Exception Destination Settings

Queue Name	Exception Destination
SIB_CuramDeadMessageQueue	System
SIB_DPEnactment	SIB_DPError
SIB_DPError	SIB_CuramDeadMessageQueue
SIB_WorkflowActivity	SIB_WorkflowError
SIB_WorkflowEnactment	SIB_WorkflowError
SIB_WorkflowError	SIB_CuramDeadMessageQueue

10. Select **OK** to apply the changes.
11. Navigate to **Resources** > **JMS** > **JMS providers**.
12. Select the **Default messaging provider** link.
13. Select the **Queues** link in the **Additional Properties** section.
14. Select **New** and set the following fields:

     • **Name**: *<QueueName>*
     • **JNDI Name**: jms/ *<QueueName>*

- **Bus Name**: CuramBus
- **Queue Name**: SIB_ *<QueueName>*
- **Delivery Mode**: Persistent

Leave everything else as the default and select **OK** to apply the changes.

15. Save the changes to the master configuration as described in .

Configure the required topics

Configure the required topics

Procedure

1. Navigate to **Resources** > **JMS** > **JMS providers**;
2. Select the **Default messaging provider** link;
3. Select the **Topics** link in the **Additional Properties** section;
4. Select **New** and set the following fields:

- **Name**: CuramCacheInvalidationTopic
- **JNDI Name**: jms/CuramCacheInvalidationTopic
- **Description**: Cache Invalidation Topic
- **Bus name**: CuramBus
- **Topic space**: Default.Topic.Space
- **JMS Delivery Mode**: Persistent

Leave everything else as the default and select **OK** to apply the changes.

5. Save the changes to the master configuration as described in .

Configure the required queue activation specifications

Configure the required queue activation specifications

About this task

As with the setting up of queues, perform these steps, substituting *<QueueName>* (without the angle brackets) with each of the following queue names: DPEnactment, DPError, CuramDeadMessageQueue, WorkflowActivity, WorkflowEnactment and WorkflowError.

Procedure

1. Navigate to **Resources** > **JMS** > **JMS providers**;
2. Select the **Default messaging provider** link;
3. Select the **Activation specifications** link in the **Additional Properties** section;
4. Create a new specification by selecting **New** and set the following fields:

- **Name**: <QueueName>
- **JNDI name**: eis/ *<QueueName>* AS
- **Destination Type**: Queue

- **Destination JNDI name**: jms/ *<QueueName>*
- **Bus Name**: CuramBus
- **Authentication Alias**: Same as for the `jdbc/curamdb` data source (e.g. *<SERVERNAME>* /dbadmin)

Leave everything else as the default and select **OK** to add the port.

Results

Save the changes to the master configuration as described in <u>Save the master configuration on page 34</u>.

Configure the required topic activation specifications

Use the steps in the procedure to configure the required topic activation specifications.

About this task

When you update the Curam cache invalidation activation specification, an error in the IBM® WebSphere® Application Server user interface sets the `durableSubscriptionHome` property to a value. The behavior is incorrect because the configuration requires that the property is blank. Therefore, each time that you update the Curam cache invalidation activation specification, you must verify that the `durableSubscriptionHome` property has a blank value in the custom properties of the activation specification, as indicated in the procedure.

Procedure

1. As with the queue activation specifications in the previous section, add a new activation specification and set the following fields:

    - **Name**: CuramCacheInvalidationTopic
    - **JNDI name**: eis/CuramCacheInvalidationTopicAS
    - **Destination Type**: Topic
    - **Destination JNDI name**: jms/CuramCacheInvalidationTopic
    - **Bus Name**: CuramBus
    - **Authentication Alias**: Same as for the `jdbc/curamdb` data source (e.g. *<SERVERNAME>* /dbadmin)

2. Leave everything else as the default value and click **OK** to apply the changes.

3. Save the changes to the master configuration, as described in <u>Save the master configuration on page 34</u>.

Verify that the `durableSubscriptionHome` property is blank:

> **Note:** Complete the following steps each time that you update the Curam cache invalidation activation specification.

4. Log on to the WebSphere® Application Server administrative console.

5. Click **Resources** > **Resource Adapters** > **J2C activation specifications**.

6. Click **CuramCacheInvalidationTopic**.

7. Click **J2C activation specification custom properties**.

8. Click `durableSubscriptionHome`.

9. Delete any value that is displayed for the property.

   The configuration requires that the property is blank.

10. In **System Administration**, click **Save Changes to Master Repository**.

11. Select the **Synchronize changes with Nodes** checkbox.

12. Restart the application servers.

Configure historical log files

Configure the maximum number of historical log files maintained by a server.

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**

2. Select the relevant server from the list of servers

3. Select **Logging and Tracing** from the **Troubleshooting** section

4. Select **JVM Logs** from the **General Properties** list

5. Change the **Maximum Number of Historical Log Files** field to 30 for both the *System.out* and *System.err* files

6. Select **OK** to apply the changes

7. Save the changes to the master configuration.

Post configuration tasks

Create database tables required for the Service Integration Bus and create the database tables required for the Timer Service.

Service Integration Bus database tables

Manually create database tables required for the Service Integration Bus (SIB). IBM® WebSphere® Application Server provides the SIB DDL Generator to generate the SQL for creating these tables.

Run the generator by executing the following command (example for Windows):

```
%WAS_HOME%\bin\sibDDLGenerator.bat
  -system system
  -platform platform
  -schema username
  -database database_name
  -user username
  -statementend ;
  -create
```

Where

- *system* is the database that is to be used, e.g. `oracle` or `db2`;
- *platform* is the operating system, such as `windows`, `unix` or `zos`;
- *username* is the username required for accessing the database, as specified in the *Bootstrap.properties* property `curam.db.username`;
- *database_name* is the name of the database to be used, as specified in the *Bootstrap.properties* property `curam.db.name`.

For example:

```
c:/Websphere/AppServer/bin/sibDDLGenerator.bat
        -system db2 -platform windows
        -schema db2admin -database curam -user db2admin
        -statementend ; -create
```

This command outputs some SQL statements which you run on the target database.

Timer service database tables

Manually create the database tables required for the Timer Service. IBM® WebSphere® Application Server provides the DDL for these tables in it's *WAS_HOME /Scheduler* directory.

The DDL files you run are the *createTablespaceXXX.ddl* and *createSchemaXXX.ddl* in that order, where *XXX* is your target database product name.

Each DDL file contains instructions appropriate for running against your target database.

Completion

The application server is now configured and ready to install an Cúram application on it. Log out of the Administration Console and restart the server using the targets described in .

1.9 Application deployment

Use the Administration Console to install an enterprise application in IBM® WebSphere® Application Server

Install an application, EJB component, or web module using the Administrative Console.

> **Note:** Once the install has been started, the **Cancel** button must be used to exit if the installation of the application is aborted. It is not sufficient to simply move to another Administrative Console page without first clicking **Cancel** on an application installation page.

1. Navigate to **Applications** > **New Application**;
2. Select **New Enterprise Application**;
3. Click the appropriate radio button and specify the full path name of the source application file or EAR file, optionally via the **Browse** button, in the **Path to the new application** panel and click **Next**;

   The default location for the application EAR files is:

   %SERVER_DIR%/build/ear/WAS/Curam.ear

4. Select the **Fast Path - Prompt only when additional information is required** radio button in the **How do you want to install the application?** panel and click **Next**;
5. Leave the defaults as they are for step 1, *Select installation options* and click **Next**;

6. In step 2, **Map modules to servers**, for every module listed, select a target server or a cluster from the **Clusters and Servers** list. To do this, tick the check box beside the particular module(s) and then select the server or cluster and click **Apply**.

7. For the following step(s) click **Next** and then **Finish** to complete the installation. This step may take a few minutes and should finish with the message *Application Curam installed successfully*.

8. Save the changes to the Master Configuration. (See Save the master configuration on page 34 for more details.)

9. Navigate to **Applications** > **Application Types** > **WebSphere enterprise applications** and select the newly installed application.

10. Select the **Class loading and update detection** option from the **Detail Properties** section.

11. Set the **Class loader order** to be **Classes loaded with local class loader first (parent last)**.

12. Set the **WAR class loader policy** to be **Single class loader for application**.

13. Click **OK**.

14. Navigate to **Users and Groups -> Manage Users**. Click **Create...** and enter a User ID, Password, First Name and Last Name. Then click on **Create**.

    See 1.5 Deploying an application on page 23 for information regarding the credentials expected here by the application and changing them.

15. Return back to the enterprise application (**Applications** > **Application Types** > **WebSphere enterprise applications**, select the newly installed application) and select the **Security role to user/group mapping** option from the **Detail Properties** section and map the mdbuser role to a username and password as per these steps:

    > **Note:** The username you use to map to the mdbuser role must already be defined in your user registry.

    1. Check **Select** for the mdbuser role and click **Map Users...**;
    2. Enter the appropriate username in the **Search String** field and click **Search**;
    3. Select the ID from the **Available:** list and click **>>** to add it to the **Selected:** list and click **OK**.
    4. Click **OK**.

16. Having mapped the mdbuser role you can now update the user RunAs role by selecting the **User RunAs roles** option from the **Detail Properties** section.

17. Enter an appropriate username and password in the **username** and **password** fields, respectively. Check **Select** for the mdbuser role and click **Apply**.

18. Click **OK**.

19. Save the changes to the master configuration.

20. After deployment it is necessary to start the application before it can be used. Navigate to **Applications** > **Application Types** > **WebSphere enterprise applications**, tick the check box for the newly installed application, and click the **Start** button. This step may take a few minutes and should finish with the application status changing to indicate it has been started.

21. Finally, test the application deployment. For example, point a Web browser at the URL for the deployed application e.g. *https://localhost:9044/Curam*.

1.10 Network deployment

IBM® WebSphere® Application Server network deployment offers advanced deployment services, including clustering, edge services and high availability for distributed configurations.

For more information see the *Cúram Third Party Tools Installation Guide* for information on the installation of WebSphere Network Deployment.

Creating profiles

Create at least two profiles, one to act as the deployment manager for the node and the others as the federated servers.

After you install WebSphere Network Deployment, create at least two profiles. One profile acts as the deployment manager for the node, the others act as the federated servers.

Crest the profiles by using the Profile Creation Wizard, which you start the *pct<hardware platform>* file from the *bin/ProfileCreator* directory of the WebSphere Application Server installation.

The first choice when using this wizard is to either create:

1. A deployment manager profile;
2. An Application Server profile.

The second is the choice to enable administrative security. Enable administrative security on profile creation. These settings can be changed later.

Federating a node

Start the targeted Deployment Manager before federating an application server profile.

Run the following command to start the Deployment Manager from the *profiles/ <deployment manager profile name>/bin* directory of the WebSphere Network Deployment installation:

startServer dmgr

To add your application server profile to the Deployment Manager node, run the following command from the *profiles/<Application Server profile name>/bin* directory of the WebSphere Application Server installation:

addNode <deploymgr host> <deploymgr port>

Where the *<deploymgr host>* and *<deploymgr port>* are the listen host and port for the Deployment Manager 's SOAP Connector. The SOAP Connector details can be found in the Deployment Manager Administrative Console under:

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**;
2. Select the relevant server from the list;
3. Expand **Ports** in the **Communications** section and press the **Details** button;

4. The required details are listed as the **Host** and **Port** for the
   SOAP_CONNECTOR_ADDRESS.

Configuring a node

Configure the server before deploying an application on the registered node by using the Deployment Manager Administration Console. The configuration is then synchronized with the node's federated servers.

The Node Agent, which enables communication between the Deployment Manager and its federated servers, is required to be started. This must be done via the *startNode.bat* or *startNode.sh* command in the *profiles/<federated profile name>/bin* directory of the IBM® WebSphere® Application Server installation.

After the Node Agent is started, all control is handed over to the Deployment Manager for this Node's servers. To start or stop a server in the Deployment Manager Administration Console:

1. Navigate to **Servers** > **Server Types** > **WebSphere application servers**;
2. Check the server to be started or stopped from the list and click the **Start** or **Stop** button as required.

Now configure the federated servers, all configuration is done through the Deployment Manager Administrative Console. 1.8 Manual WebSphere® Application Server configuration on page 26 describes the manual WebSphere® Application Server configuration for a basic installation, and should be followed with the differences identified below. When saving the master configuration ensure you manually force synchronization via the Administrative Console:

1. Navigate to **System Administration** > **Save Changes to Master Repository**;
2. Check the **Synchronize changes with Nodes** check box;
3. Select the **Save** button. The synchronization may take some time;
4. Select the system or WebSphere® Application Server logs for synchronization completion. These messages might vary by WebSphere® Application Server release, but you are looking for something like:

   ADMS0208I: The configuration synchronization complete for cell.

   Once synchronization is complete, review the server status and various logs to ensure success;

Configure administration security on page 34 details the security setup required during manual configuration. This setup requires the *Registry.jar* to be copied to a directory within the WebSphere® Application Server installation. The *Registry.jar* should be copied from *CuramSDEJ/lib* to the *lib* directory of the Deployment Manager installation and any federated installations.

Configure administration security on page 34 this security setup also requires the *CryptoConfig.jar* to be copied to the *java/jre/lib/ext* directory within the WebSphere Application Server installation. The *CryptoConfig.jar* should be copied to the same directory structure for any other WebSphere Application Server installations in the environment.

> **Note:** Before building the `Curam.ear` for deployment note the *BOOTSTRAP_ADDRESS* of the server that these are installed onto. The *BOOTSTRAP_ADDRESS* is located in the same list of ports as the *SOAP_CONNECTOR_ADDRESS* described previously.
>
> By default the *BOOTSTRAP_ADDRESS* expected by the application is 2809. To solve this issue either change this address or alternatively change the relevant property in your `AppServer.properties` file.
>
> The property that should be changed is the curam.server.port value in the `AppServer.properties` file. Changing this affects the port value in the `web.xml` file when building an EAR file. For more information on the `web.xml` file consult the *Cúram Web Client Reference Manual*.

Deploying on the Node

Finally, should be followed to manually deploy the applications on the required server. Applications can then be started or stopped using the Deployment Manager Administration Console.

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.