IBM Cúram Social Program Management

Cúram Deployment Guide for WebSphere Application Server

Version 6.0.4

Note

Before using this information and the product it supports, read the information in Notices at the back of this guide.

This edition applies to version 6.0.4 of IBM Cúram Social Program Management and all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Chapter 1

Introduction

1.1 Deployment Guide

This guide discusses the steps necessary to build a *IBM® Cúram Social Program Management* application for deployment on the base version of *IBM® WebSphere® Application Server* [1]. The guide also details the support provided for configuring and deploying on *WebSphere Application Server*, and where necessary provides the manual steps required.

It is a prerequisite that the reader have knowledge of the *IBM Cúram Social Program Management* 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. For details on this installation consult the *Cúram Third Party Tools Installation Guide* [2].

Notes

[1]For information on using the application with the *Network Deployment* edition of *WebSphere Application Server* consult Appendix A, *Manual WebSphere Application Server Configuration*.
[2]Refer to the installation guide that is relevant to the platform, i.e. *Microsoft® Windows* or UNIX.

Chapter 2

Building EAR Files

2.1 Introduction

The main step before deployment of a *IBM Cúram Social Program Management* 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 following environment variable is set:

- `WAS_HOME`

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

2.2 The Enterprise Application

2.2.1 Building the Application EAR File

The following target should be executed from the root directory of the server project to create the application EAR file for *WebSphere Application Server*:

build websphereEAR

This target will create a ready to install EAR file, `<SERVER_MODEL_NAME>.ear` located in `<SERVER_DIR>/build/ear/WAS`[1].

This target will also create a ready to install `.ear` file, `CuramHelp.ear`, located in `<SERVER_DIR>/build/ear/WAS`, provided the online help application has been created and built. For details on how to build the online help application, please refer to the *Cúram Online Help System Development and Deployment Guide*. For details on the contents of the online help file, please refer to Section 2.3.2, *Contents of the CuramHelp.ear File*

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

Note

An EAR file cannot be built for *H2* database. [2]

2.2.2 Under the Hood

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, i.e., the methods of *Facade* classes or `<<WebService>>` classes and can be called by remote clients.

By default all remote calls to the server are handled by the session bean `curam.util.invoke.EJBMethod`, rather than a session bean per publicly available interface. This bean provides support for *IBM Cúram Social Program Management* features such as authorization, auditing and tracing. If required it is also possible to generate a Facade interface[3].

2.2.3 Contents of Application EAR File

The EAR file that is produced has the following structure and contents:

- META-INF Directory
  - `application.xml`

    This file is automatically generated and 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`

    WebSphere Application Server security policy file that grants the application the *Java* permission `java.security.AllPermission`.
  - `MANIFEST.MF`

The manifest file which details the contents of the EAR file.

- **Core JAR Files.** The core JAR files include[4]:
  - `antlr.jar`
  - `appinf.jar`
  - `appinf_internal.jar`
  - `coreinf.jar`
  - `rules.jar`
  - `jde_commons.jar`
  - `log4j.jar`
  - `commons-pool.jar`
  - `commons-codec.jar`
  - `commons-discovery.jar`
  - `jdom.jar`
  - `axis.jar`
  - `castor.jar`
  - `jaxrpc.jar`
  - `saaj.jar`
  - `java_cup.zip`
  - `InfrastructureModule.jar`
  - `InvalidationModule.jar`
  - `DBtoJMS.war`
  - `ClientModule.war`
- **Facade JAR Files.** This is only present if facade generation has been enabled. All facades defined in the application are packaged into one JAR file, `FacadeModule.jar`. This JAR file contains the bean implementation classes for the *EJB* modules that represent the facades. The JAR file contains the following files in the `META-INF` directory:
  - `ejb-jar.xml`

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

- `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.

2.3 The Online Help Application

The *IBM Cúram Social Program Management* online help application is built in a separate EAR file (e.g. `CuramHelp.ear`). The `CuramHelp.ear` file automatically gets generated when the application EAR file is created. Please refer to Section 2.2.1, *Building the Application EAR File* for further details on building the Application EAR file. The ability to build the online help application EAR file separately is also available.

2.3.1 Building the CuramHelp.ear File

The following target should be executed from the root directory of the server project to create the application `CuramHelp.ear` file for *WebSphere Application Server*:

build websphereHelpEAR

This target will create a ready to install `CuramHelp.ear` file, located in the `<SERVER_DIR>/build/ear/WAS` directory, provided the online help application has been created and built. For details on how to build the online help application, please refer to the *Cúram Online Help System Development and Deployment Guide*.

2.3.2 Contents of the CuramHelp.ear File

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

- **META-INF Directory.** The META-INF Directory includes the following:

    - `application.xml`

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

    - `MANIFEST.MF`

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

- **help.war files.** A help.war is a web application containing application online help screens. A help.war will be created for each locale supported. Please refer to the *Cúram Online Help System Development and Deployment Guide* for further details on online help.

2.4 The Web Services Application

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

2.4.1 Building the Web Services EAR File

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

build 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 *IBM Cúram Social Program Management* 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 will create 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 *IBM Cúram Social Program Management* application, ready for deployment, must exist.

2.4.2 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[5]. Any server component with a stereotype of `<<webservice>>` will be treated as if it also had a stereotype of `<<ejb>>`. This is because Web Service interfaces are wrappers on publicly available BPOs.

2.4.3 Contents of Web Services EAR 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* permission `java.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 *IBM Cúram Social Program Manage-*

ment 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 *IBM Cúram Social Program Management* BPOs to web services.

2.4.4 Web Service WSDL

A *IBM Cúram Social Program Management* 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 will be at the *URL*:

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

The *URL*

`http://localhost:9082/CuramWS/services`

will return 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

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

2.5 Multiple EAR files

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

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

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

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

Example 2.1 deployment_packaging.xml sample

Each file can have multiple `ear` elements and results in an EAR file being produced in the `SERVER_DIR/build/ear/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 *IBM Cúram Social Program Management* server module should be deployed per cluster. If `requireServer` is set to `true` for multiple EAR files, then the other EAR files must be deployed in another cluster to avoid conflicts.

- `components`

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

- `context-root`

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

- `custom-web-xml`

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

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

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

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

2.6 Alternative Targets

The **websphereEAR** target will build an *IBM Cúram Social Program Management* application `.ear` file containing both the web client and application. Support is provided to build an application `.ear` file containing only the web application or only the server application.

These targets may be necessary where the web client and server application are required 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. [6]

To build an `ear` file containing only the web client application the following command should be used:

build websphereEAR -Dclient.only=true

To build an `.ear` file containing only the server application the following command should be used:

build websphereEAR -Dserver.only=true

Notes

[1]SERVER_MODEL_NAME and SERVER_DIR are environment variables which specify the name of the model in the project and the root directory of the project respectively.
[2]For more information on *H2* database consult the *Cúram Third-Party Tools Installation Guide for Windows*.
[3]The optional build parameter `-Denablefacade=true` turns on the generation of facade code.
[4]The version numbers are not listed for the JAR files detailed.
[5]Consult the *Cúram Server Modelling Guide* for details on assigning BPOs to server components.
[6]For more information on External Access Security consult the *Cúram Server Developers Guide*.

Chapter 3

Application Server Configuration

3.1 Introduction

This chapter presumes that *WebSphere* has already been installed. Consult the *Cúram Third Party Tools Installation Guide*[1] for details on the installation.

The configuration of *WebSphere Application Server* is similar on all platforms and the *Server Development Environment for Java* (SDEJ) provides a number of *Ant* targets to aid the configuration and management of the installation. For those interested, Appendix A, *Manual WebSphere Application Server Configuration* details the manual steps performed by the configuration scripts.

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

Note

The **configure** target will overwrite the *default* profile created by *WebSphere Application Server* unless `-Dkeep.profile=true` is passed to the target.

3.2 WebSphere Application Server Configuration

The configuration of *WebSphere Application Server* involves setting up a profile, data source, a number of servers and configuring the JMS and security settings. All these tasks can be performed by executing the **configure** target provided by the *SDEJ*.

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

- `profile.name=AppSvr01`

- `cell.name=${node.name}Cell`

The command **build configure** should be executed from the `<SERVER_DIR>` directory to invoke automatic configuration. This target requires that the files `AppServer.properties` and `Bootstrap.properties` exist in the `<SERVER_DIR>/project/properties` [2] directory. See the *Cúram Server Developer's Guide* for more information on the setup of a `Bootstrap.properties`. Example 3.1, *AppServer.properties.sample* shows example contents of the `AppServer.properties` file.

```
## APPLICATION SERVER PROPERTIES

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

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

# 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
```

Example 3.1 AppServer.properties.sample

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 Example 3.1, *AppServer.properties.sample* example and set by the configura-

tion 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. 3

3.3 Security Configuration

The default security configuration of *IBM Cúram Social Program Management* within *WebSphere Application Server* involves the default file-based user registry and a JAAS Login Module. The *Default Configuration for IBM WebSphere Application Server* section in the *Cúram Security Handbook* should be referenced for further details on this.

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.

3.3.1 Special Configuration Steps When Using Identity Only and LDAP

When using identity only in combination with *WebSphere Application Server* and LDAP you may need to perform additional manual configuration steps; this is regardless of whether configuration is done via the *WebSphere Application Server* administrative console or the `configure` target. With this combination you may find that *WebSphere Application Server* fails to start successfully and this is due to the need to add a *WebSphere Application Server*-generated username to the login module exclude list property (`ex-`

`clude_usernames`) described in Section A.2.10.1, *Add the Login Module*. In this case of *WebSphere Application Server* failing to start there will be a SECJ0270E error message in the `SystemOut.log` file prior to the failure.

These are the steps needed to resolve this error:

1. Identify the username that is causing *WebSphere Application Server* start to fail. Configure the login module trace as described in Section 3.3.3, *Logging the Authentication Process* (in regard to the `configure` target) or Section A.2.10.1, *Add the Login Module* (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*.

3.3.2 WebSphere Application Server User Registry

By default the configured *WebSphere Application Server* user registry is not queried as part of authentication. When the login module is configured for identity only, the user registry is queried. It is possible to override this default behavior by setting the `curam.security.user.registry.enabled` property. If this property is set to `true` the *WebSphere Application Server* user registry will be queried during the authentication process, regardless of whether identity only authentication is enabled or disabled. If this property is set to `false`, the *WebSphere Application Server* user registry will not be queried. For example, if `curam.security.check.identity.only` is set to `true` and `curam.security.user.registry.enabled` is set to `false`, neither the *IBM Cúram Social Program Management* authentication verifications nor the *WebSphere Application Server* user registry will be used as part of the authentication process.

You can also control the authentication of types of external users (i.e. non-internal users) against the *WebSphere Application Server* user registry via use of the `curam.security.user.registry.enabled.types` and/or the `curam.security.user.registry.disabled.types` properties. These properties specify a comma-delimited list of external user types that will, or will not be, authenticated via the *WebSphere Application Server* user registry:

- User types specified in the `curam.security.user.registry.enabled.types` list will be processed against the *WebSphere Application Server* user registry (e.g. LDAP) and your `ExternalAccessSecurity` implementation.
- User types specified in the `curam.security.user.registry.disabled.types` list will not be processed against the *WebSphere Application Server* user registry and the processing of your `ExternalAccessSecurity` implementation will be the authority for authentication.

The precedence order in processing these three properties and the *WebSphere Application Server* user or external (e.g. LDAP) registry is as follows:

- By default the *WebSphere Application Server* user registry is not checked and application authentication is used.

- The setting of the `curam.security.user.registry.enabled` property to `true` requires authentication by both the *WebSphere Application Server*, or external (e.g. LDAP), user registry and application security (for internal users) or your `ExternalAccessSecurity` implementation (for external users).
- An external user of a type specified in the `curam.security.user.registry.enabled.types` list must be authenticated by the *WebSphere Application Server*, or external, user registry and your `ExternalAccessSecurity` implementation.
- An external user of a type specified in the `curam.security.user.registry.disabled.types` list is not authenticated by the *WebSphere Application Server*, or external, user registry and your `ExternalAccessSecurity` implementation is the authority.

See Section A.2.10, *Set up the System JAAS Login Module* for more information on setting the resultant properties in the `CuramLoginModule` configuration.

3.3.3 Logging the Authentication Process

`curam.security.login.trace` is an optional property that will enable logging for the login module. When set to `true` this property results in tracing information being added to the *WebSphere Application Server* `SystemOut.log` file during the authentication process.

3.3.4 Establishing an Alternate Exclude Username Delimiter

`curam.security.usernames.delimiter` is an optional property that will enable setting an alternate delimiter for the list of usernames in the `exclude_usernames` property. The property can be set to a character that will allow usernames with embedded commas such as with LDAP.

3.3.5 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 Application Server Caching Behavior* section in the *Cúram Security Handbook* should be consulted for further details on this.

3.3.6 Security custom properties

- `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 Section A.2.10, *Set up the System JAAS Login Module* 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.

3.3.7 Security hardening measures

When a user logs into the application, they provide a username & password. This is sent to the server, and if successfully authenticated, the server responds with a unique token. The token, in this case, is 'LTPA token'. This token is used in all subsequent requests to recognize the user and then serves privileged content. When the user logs out, we would expect this token to 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 Section A.2.7, *Configure Administration Security*.

For more information see:

- ht-tp://www.ibm.com/developerworks/websphere/techjournal/1004_botzum/1004_botzum.html?ca=drs#step19

- ht-tp://www.ibm.com/developerworks/websphere/techjournal/1004_botzum/1004_botzum.html?ca=drs#step29

3.4 Time Zone Configuration

If multiple server machines are used, they all must have their clocks in sync and be in the same time zone in order that the "natural" ordering of date/times on the database accurately reflects the order that the events occurred in the real world. For example if on the database record *A* has a creation date/time field earlier than that on record *B*, then we can say for sure that *A* was created before *B*, no matter which server created either record.

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

3.5 Starting and Stopping WebSphere Servers

A number of *Ant* targets are provided to aid in the starting and stopping of *WebSphere* servers. These targets should be executed from the `<SERVER_DIR>` directory and as for the **configure** target, they require the `AppServer.properties` file to be setup correctly (Example 3.1, *AppServer.properties.sample*). They also require a number of extra parameters to be specified and these are detailed below.

3.5.1 Start a WebSphere Server

The target for starting a *WebSphere* server is **startserver** and requires the following options:

- `-Dserver.name`

  The name of the server to be started.

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.

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

Example 3.2 Example of Usage

3.5.2 Stop a WebSphere Server

The target for stopping a *WebSphere* server is **stopserver** and requires the following options:

- `-Dserver.name`

  The name of the server to be stopped.

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

Example 3.3 Example of Usage

3.5.3 Restart a WebSphere Server

The target for restarting a *WebSphere* server is **restartserver** and the options are the same as for the **startserver** target. See Example 3.2, *Example of Usage* for an example of usage.

Note

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

Notes

[1]Refer to the installation guide that is relevant to the platform, i.e *Windows* or `UNIX`.
[2]It is possible to override this default location for the properties file by specifying

`-Dprop.file.location=<new location>` when executing the **configure** target.
[3]For more information on *H2* database consult the *Cúram Third-Party Tools Installation Guide for Windows*.

Chapter 4

Deployment

4.1 Introduction

The final step after packaging the *IBM Cúram Social Program Management* application and web services application in EAR files is to deploy them to the application server.

Before deploying, it is important to note that in *WebSphere Application Server* the configuration scripts provided support a simple configuration targeted at the single server in the *Express* or *Base* editions of *WebSphere Application Server*.

4.2 Deployment

The *SDEJ* provides targets for installing and un-installing applications on a *WebSphere* server. As with the **startserver**/ **stopserver** targets, the **installapp**/ **uninstallapp** targets require that the `AppServer.properties` file is configured correctly (see Example 3.1, *AppServer.properties.sample*). The targets also require a number of options to be specified and these are detailed below.

Ensure the server is started before installing an application. There is no need to restart the server after installation, as the install target will automatically start the application.

4.2.1 Install an 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.

```
build installapp -Dserver.name=CuramServer
    -Dear.file=d:/ear/Curam.ear
    -Dapplication.name=Curam
```

Example 4.1 Example of Usage

Note

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

4.2.2 Change SYSTEM Username

It is strongly recommended that you change the username for JMS invocation while deploying the application. The following properties should be set in the `AppServer.properties` file before deployment to modify this username:

- `runas.user`

  The username JMS invocations should run under.

- `runas.password`

  The encrypted password associated with the username. The password should be encrypted using the **encrypt** target. See the *Cúram Server Developers Guide* for more information.

It is also possible to change the username once the application has been deployed using the *WebSphere Application Server* administrative console. Navigate to *Applications*→*Application Types*→*WebSphere enterprise applications* and select the *IBM Cúram Social Program Management* application. Select the *User RunAs roles* link. Check the `everyone` role, enter a new username and password (note, password should be entered in the unencrypted format here) and click the *Apply* button. Save the changes as detailed in Section A.2.6, *Save the Master Configuration*.

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

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

4.2.3 Uninstall an Application

The *Ant* target to uninstall an application is **uninstall** and requires the following options:

- `-Dserver.name`

  The name of the server the application is installed on.

- `-Dapplication.name`

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

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

Example 4.2 Example of Usage

4.3 Pre-compiling JSPs

There is one additional target available during deployment, **precompilejsp**, which allows for the *JSP*s of a client EAR to be pre-compiled *before* installing the EAR file. Pre-compiling the *JSP*s before installation will speed up the display of a particular 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=d:/Curam.ear
```

Example 4.3 Example of Usage

Note

While running the **precompilejsp** target for *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 the `JspBatchCompiler.bat` script in the `%WAS_HOME%\bin` directory should be modified to increase the maximum memory size. Change the memory consumption from `-Xmx256m` to at least `-Xmx1024m`.

4.4 Testing Deployment

When the application is installed[1] on a configured *WebSphere Application Server* installation the next step is to start and test the application.

Ensure the relevant server is started[2] 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 Section 2.5, *Multiple EAR files*, for details).

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.

Notes

[1]The installation of a web services application may also be required.
[2]There is no need to restart the server after an application is deployed.

Appendix A

Manual WebSphere Application Server Configuration

A.1 Introduction

The sections of this chapter cover the manual steps required to configure and deploy on the *Base* or *Express* edition of *WebSphere Application Server*. You will have to alter these steps appropriately to deploy in a *Network Deployment* installation of *WebSphere Application Server*. See Section A.4, *WebSphere Network Deployment* for more information in this area.

A.2 Manual WebSphere Application Server Configuration

The *IBM WebSphere Application Server* installation can be configured manually if required, but this is not recommended. This section details the manual steps required to configure *WebSphere Application Server* for information purposes only.

It is worth noting that any settings entered under the *Resources* section of the *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, i.e. 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.

A.2.1 The Administrative Console

Most of the configuration of *WebSphere Application Server* is done using the *Administrative Console*. To run the *Administrative Console*, the default server, e.g. `"server1"`, must be started as the *Administrative Console* is installed as a web application on this server.

To start `"server1"`, the `startServer.bat`, located in the `profiles/AppSvr01/bin` directory of the *WebSphere Application Server* installation, should be used:

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

To open the *Administrative Console*, a web browser should be pointed at:

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

Alternatively, the *Administrative Console* can be started from *Start→Programs→IBM WebSphere→Application Server V7.0→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 will be requested for login. 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 to '`Navigate to`', the tree hierarchy should be traversed to the relevant node.

A.2.2 Scripting Support

To support the execution of provided *Ant* scripts it is necessary to change the *WebSphere Application Server* property files.

sas.client.props

Open the `sas.client.props` file found in the `profiles/AppSvr01/properties` directory of the *WebSphere Application Server* installation. It is necessary to set the login source to retrieve the username and password from a properties file rather than having to type them in each time the scripts are run. 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
com.ibm.CORBA.principalName=curam
```

where *`websphere`* is the username and password for the *Administration Console*.

soap.client.props

Open the `soap.client.props` file, also found in the `profiles/AppSvr01/properties` directory of the *WebSphere Application Server* installation. It is necessary to set the login source to retrieve the username and password from a properties file rather than having to type them in each time the scripts are run. Set the following properties to be:

```
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 set to be at least:

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

server.policy

Open the `server.policy` file found in the `profiles/AppSrv01/properties` directory of the *WebSphere Application Server* installation. 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 *`<profile.name>`* is the name of the target *WebSphere Application Server* profile;

where *`<cell.name>`* is the name of the target *WebSphere Application Server* cell;

where *`<SERVER_MODEL_NAME>`* is the name of the application (base name of the EAR file).

A.2.3 Creating the Data Source Login Alias

IBM® DB2®, *IBM DB2 for z/OS®*, and *Oracle® Database* are the databases supported. The *Administrative Console* can be used to configure a login alias for both the *DB2* and *Oracle* data sources as follows:

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.

A.2.4 Configure DB2 Data Sources

Set up DB2 Environment Variable

1. Navigate to *Environment→WebSphere variables*;
2. Select the `DB2UNIVERSAL_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/Type 2 drivers. This is normally the `drivers` directory under the *SDEJ* installation, e.g. `D:\Curam\CuramSDEJ\drivers`;
4. Click *OK* to confirm the changes.

Set up the Database Driver Provider

1. Navigate to *Resources→JDBC→JDBC providers*;
2. *Note:* The appropriate scope where the data source is to be defined should be selected at this point.
3. Click *New* to add a new driver. This will open a configuration screen;
4. Select *DB2* from the list in the *database type* drop down supplied;
5. Select the *DB2 Universal JDBC Driver Provider* from the list in the *Provider type* drop down supplied;
6. Select the *XA data source* from the list in the *Implementation type* drop down supplied;
7. Click *Next* to continue;
8. Review the properties on the configuration screen that opens. There should be no need to change any of them unless you are planning to

connect to a `zOS` database. If so, verify that `${DB2UNIVERSAL_JDBC_DRIVER_PATH}` field is pointing at the correct directory for your system. For example, it should point at the directory containing the *DB2 Connect* license jar, `db2jcc_license_cisuz.jar` provided by IBM for `zOS` connectivity;

9. Click *Next* and then *Finish* to confirm the changes.

Set up the Database Driver DataSource

The following steps should be repeated for each of the Data Sources, substituting `curamdb`, `curamsibdb` and `curamtimerdb` for `<DatasourceName>` (without the angle brackets):

1. Select the `DB2 Universal JDBC Driver Provider (XA)` now displayed on the list of *JDBC Providers*. This will open the configuration screen for the provider;
2. Select the *Data sources* link under *Additional Properties*;
3. Click *New* to add a new data source;
4. Set the fields as follows:

   Data source name: `<DatasourceName>`

   JNDI name: `jdbc/<DatasourceName>`
5. Click *Next* to continue;
6. Set the fields as follows:

   Driver type: 2 or 4 as required;

   Database name: The name of the *DB2* database;

   Server name: The name of the *DB2* database server;

   Port number: The *DB2* database server port;

   Leave all other fields untouched unless a specific change is required and click *Next*;
7. 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 one set up in Section A.2.3, *Creating the Data Source Login Alias*;

   Leave all other fields untouched unless a specific change is required and click *Next* to continue.

8. Click *Finish* to confirm the changes and continue;
9. Select the newly created `DatasourceName` data source from the displayed list;
10. Select the *Custom Properties* link under *Additional Properties*;
11. Select the `fullyMaterializeLobData` entry;
12. Set the `value` to be `false`;
13. Click *OK* to confirm the change.

A.2.5 Configure an Oracle Data Source

Set up Oracle Environment Variable

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. Click *OK* to confirm the changes.

Setup the XA Database Driver

1. Navigate to *Resources→JDBC→JDBC providers*;
2. *Note:* The appropriate scope where the data source is to be defined should be selected at this point.
3. Click *New* to add a new driver. This will open a configuration screen;
4. Select the *Oracle* from the list in the *Database type* drop down supplied;
5. Select the *Oracle JDBC Driver* from the list in the *Provider type* drop down supplied;
6. Select the *XA data source* from the list in the *Implementation type* drop down supplied;
7. Set the *Name* field to be Oracle JDBC Driver (XA), if not filled in automatically;
8. Click *Next* to continue;
9. Review the *Class path* and ensure the `OR-`

ACLE_JDBC_DRIVER_PATH environment variable is correct. Click *Next* to continue;

10. Review the properties on the configuration screen that opens. There should be no need to change any of them;
11. Click *Finish* to confirm the changes.

Setup the Non-XA Database Driver

1. Navigate to *Resources→JDBC→JDBC providers*;
2. Click *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. Click *Next* to continue;
8. Review the *Class path* and ensure the `ORACLE_JDBC_DRIVER_PATH` environment variable is correct. Click *Next* to continue;
9. Review the properties on the configuration screen that opens. There should be no need to change any of them;
10. Click *Finish* to confirm the changes.

Set up the XA Database Driver DataSources

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

1. Select the `Oracle JDBC Driver (XA)` now displayed on the list of existing providers. This will open the configuration screen again;
2. Select the *Data sources* link under *Additional Properties*;
3. Click *New* to add a new data source;
4. Set the fields as follows:

   Data source name: *`<DatasourceName>`*

   JNDI Name: *`jdbc/<DatasourceName>`*

Click *Next*;

5. 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; and

   `databaseServiceName` is the service name of the database.

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

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

6. Set the *Authentication alias for XA recovery*: <*`valid for database`*>

   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 Section A.2.3, *Creating the Data Source Login Alias*;

   Leave all other fields untouched unless a specific change is required and click *Next*;

7. Click *Finish* to confirm the changes and continue.

Set up the Non-XA Database Driver DataSource

1. Select the `Oracle JDBC Driver` now displayed on the list of existing providers. This will open the configuration screen again;
2. Select the *Data sources* link under *Additional Properties*;
3. Click *New* to add a new data source;
4. Set the fields as follows:

   Data source name: `curamtimerdb`

   JNDI Name: `jdbc/curamtimerdb`

   Click *Next*;
5. 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.

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

   Leave all other fields untouched unless a specific change is required and click *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.
6. 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 Section A.2.3, *Creating the Data Source Login Alias*;

Leave all other fields untouched unless a specific change is required and click *Next*;

7. Click *Finish* to confirm the changes and continue.

A.2.6 Save the Master Configuration

A *Save* can be performed by clicking the *Save* link in the *Message(s)* box. This box is displayed only after configuration changes have been made.

A.2.7 Configure Administration Security

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

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. Select *Use Java 2 security to restrict application access to local resources* and *Warn if applications are granted custom permissions*;
11. Click the *Apply* button to confirm the changes;
12. Navigate to *Security→Global security*;
13. Select the *Custom Properties* link;
14. Click *New* and set the name and value as follows:

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

    Value=`true`
15. Click *OK* to add the new property.
16. Navigate to *Security→Global security*;

17. Select *Web and SIP Security→Single sign-on (SSO)*;
18. Check the *Requires SSL*check box;
19. Click *OK* to confirm the change.
20. Navigate to *Security→Global Security*;
21. Select *Custom properties*;
22. Add `com.ibm.ws.security.addHttpOnlyAttributeToCookies` with value `true`;
23. Click *OK* to confirm the change.
24. Save the changes to the master configuration.

A.2.8 Restart the Application Server

This step is compulsory. The server must be restarted for the security changes to take effect and to add additional required users. The server can be stopped using the `stopServer.bat` file in the `profiles/AppSrv01/bin` directory of the *WebSphere Application Server* installation. This is similar to starting the server described in Section A.1, *Introduction*.

Before restarting the application server (e.g. *server1*), it is necessary to make the registry JAR file available to *WebSphere Application Server*. The registry JAR file contains classes necessary for the security configuration.

`Registry.jar` is located in the `lib` directory of the *SDEJ* installation. Copy this file into the `lib` directory of the *WebSphere Application Server* installation. Now start the application server (e.g. *server1*) and open the *Administrative Console* to continue with the configuration steps.

Since the security configuration is complete and the scripting changes have been made, it is now possible to use the *SDEJ* scripts to restart the *WebSphere Application Server*. See Section 3.5, *Starting and Stopping WebSphere Servers* for more details on restarting the server.

The *Administrative Console* should now be opened to continue with the configuration. Now that global security is enabled, you will be required to login to the console with the username *`websphere`* and password *`websphere`* set up previously.

A.2.9 Configure Users

As detailed in Section 3.3, *Security Configuration*, 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.

- Navigate to *Users and Groups→Manage Users*;
- Click the *Create* button;
- Fill in the details for the *WebSphere* administrative user and click the *Create* button.
- Repeat the steps for the database user.

Note: If *WebSphere* administrative security was enabled when creating the profile the administrative user may already be defined in the registry.

A.2.10 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 below steps for each of these configurations.

Add the Login Module

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. The login module should be configured for the DEFAULT, WEB_INBOUND and RMI_INBOUND aliases;
4. Click *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 below, pressing *New* as needed.

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 users and the data-

Name	Example Value	Description
		base user. 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.
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.

Name	Example Value	Description
		Note If you are specifying identity only and using LDAP you may need to perform additional configuration steps; please see Section 3.3.1, *Special Configuration Steps When Using Identity Only and LDAP*.
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 Section 3.3.2, *WebSphere Application Server User Registry* 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 Section 3.3.2, *WebSphere Application Server*

Name	Example Value	Description
		User Registry for more information on the processing of the *WebSphere Application Server* user registry.

Table A.1 CuramLoginModule Custom Properties

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

Reorder the Login Module

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. The login module should be reordered for the DEFAULT, WEB_INBOUND and RMI_INBOUND aliases;
4. Click the *Set Order* button;
5. Select *curam.util.security.CuramLoginModule* and click the *Move Up* button. Repeat this until the `CuramLoginModule` entry is the top entry in the list;
6. Click *OK* to confirm the modifications to the order.

Disable Cross Cluster Authentication

This property determines the behavior of a single sign-on LTPA Token2 login. The property `com.ibm.ws.security.webChallengeIfCustomSubjectNotFound` is set 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.

1. Navigate to *Security*→*Global security*;
2. Click on *Custom properties* and select *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. Click *OK* to confirm the addition;

Save the Changes

Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

A.2.11 Server Configuration

Configure your JNDI lookup port

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 click the *Details* button;
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. Click *OK* to apply changes;
6. Save the changes made to the master configuration using the *Save* option as before.

Configure your ORB Pass By Reference

1. Navigate to *Servers*→*Server Types*→*WebSphere application servers*;
2. Select the relevant server from the list, e.g. server1;
3. Expand *Container Services* in the *Container Settings* section and click 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 using the *Save* option as before.

Configure your Java Virtual Machine

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: `512`

   Maximum heap size: `1024`

   Click *Apply* to set the values;
7. In the *Additional Properties* section Select the *Custom Properties* link;
8. Click *New* and set the properties as follows:

   Name: `com.ibm.websphere.security.util.authCacheCustomKeySupport`

   Value: `false`

   Click *OK* to add the property;
9. *The following step is only required on non-Windows platforms.*

   Click *New* and set the properties as follows:

   Name: `java.awt.headless`

   Value: `true`

   Click *OK* to add the property;
10. Save the changes made to the master configuration using the *Save* option as before.

Configure your Timer Service

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 Section A.2.3, *Creating the Data Source Login Alias*;

7. Click *OK* to confirm the changes;
8. Save the changes made to the master configuration using the *Save* option as before.

Set up the Port Access

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

   User-defined Port Name: `CuramClientEndPoint`

   Host: `*`

   Port: `9044`

   Click *OK* to apply the changes;
6. Click *New* and set the following fields for the WebServices TCP/IP port:

   User-defined Port Name: `CuramWebServicesEndPoint`

   Host: `*`

   Port: `9082`

   Click *OK* to apply the changes;
7. Navigate to *Servers*→*Server Types*→*WebSphere application Servers*;
8. Select the relevant server from the list;
9. Expand the *Web Container Settings* branch in the *Container Settings* section;
10. Select the *Web container transport chains* link;
11. Click *New* and set the following fields for the Client transport chain:

    Name: `CuramClientChain`

    Transport Chain Template: WebContainer-Secure

    Click *Next*

    Use Existing Port: CuramClientEndPoint

    Click *Next* and *Finish*
12. Click *New* and set the following fields for the WebServices transport chain:

    Name: `CuramWebServicesChain`

Transport Chain Template: WebContainer

Click *Next*

Use Existing Port: CuramWebServicesEndPoint

Click *Next* and *Finish*

13. Select the newly created *CuramClientChain*;
14. Select the *HTTP Inbound Channel* link;
15. Ensure the *Use persistent keep alive connections* check box is checked;
16. Click *OK* to confirm the addition;
17. Navigate to *Environment→Virtual hosts*;
18. Click *New* to add a new `Virtual Host` by setting the following fields;

    Name = *`client_host`*

    Repeat this step using the replacing *`client_host`* with *`webservices_host`*;
19. Select the *client_host* link from the list of virtual hosts;

    Select the *Host Aliases* link in the *Additional Properties* section;

    Click *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);
20. Click *OK* to confirm the addition;
21. Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

Configure Session Security Integration

1. Navigate to *Servers→Server Types→WebSphere application servers*;
2. Select the relevant server from the list;
3. Click the *Session management* in the *Container Settings* section
4. Select the *Security integration, un-check. Note: Please make sure security integration is un-checked.*
5. Click *OK* to apply changes;
6. Save the changes made to the master configuration using the *Save* option as before.

7. **Note**

   This above setting is required for *IBM Cúram Social Program Management* web applications.

A.2.12 Bus Configuration

Setup the Service Integration Bus

1. Navigate to *Service integration*→*Buses*;
2. Click *New* and in *Step 1* set the following field:

   Name: CuramBus

   Leave everything else as the default and click *Next*;
3. Entering the *Configure bus security* Wizard, Step 1.1, click *Next*;

   In *Step 1.2* of the *Configure bus security* Wizard take the default setting and click *Next*;

   In *Step 1.3* of the *Configure bus security* Wizard take the default setting, as appropriate, and click *Next*;

   In *Step 1.4* of the *Configure bus security* Wizard review your settings and click *Next*;
4. In Step 2 click *Finish* to apply the changes.
5. Select the *CuramBus* now displayed on the list of Buses. This will open the configuration screen;
6. Select *Bus members* in the *Topology* section;
7. Click *Add* to open the *Add a New Bus Member* Wizard;
8. Select the server to add to the Bus and click *Next*;
9. Select *Data store* and click *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 click *Next*;
11. Take the default tuning parameters as appropriate and click *Next*;
12. Click *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. Click *New* to open the *SIB Security Resource Wizard*;
18. Select the *The built in special groups* radio button and click *Next*;
19. Select the *Server* and *AllAuthenticated* check boxes and click *Next*;
20. Click *Finish* to complete and exit the Wizard.
21. Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

A.2.13 JMS Configuration

Setup the JMS Connection Factories

1. Navigate to *Resources*→*JMS*→*JMS providers*;
2. *Note:* The appropriate scope where the JMS resources are to be defined should be selected at this point.
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 (e.g. *<SERVERNAME>*/dbadmin)

   Mapping-configuration alias: DefaultPrinicipalMapping

   Container-managed authentication alias: Same as for the Authentication alias for XA recovery.

   Leave everything else as the default and click *OK* to apply the changes;
6. Click *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 (e.g. <*SERVERNAME*>/dbadmin)

Mapping-configuration alias: DefaultPrinicipalMapping

Container-managed authentication alias: Same as for the `jdbc/curamdb` data source (e.g. <*SERVERNAME*>/dbadmin)

Leave everything else as the default and click *OK* to apply the changes;

7. Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

Note

With the above 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 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.

   a. `$AdminConfig getid /Node:MyNode`

b. `$AdminTask showSIBJMSConnectionFactory CuramQueueConnectionFactory(cells/MyNodeCell/nodes/MyNode|resources.xml#J2CConnectionFactory_1264085551611)`

   Here 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}
```

c. `$AdminTask modifySIBJMSConnectionFactory CuramQueueConnectionFactory(cells/MyNodeCell/nodes/MyNode|resources.xml#J2CConnectionFactory_1264085551611) {-authDataAlias crouch/databaseAlias}`

d. `$AdminConfig save`

e. You can re-show the resource to verify the change.

f. Repeat the steps for the `CuramTopicConnectionFactory`.

g. Restart the application server.

Setup the Required Queues

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

1. Navigate to *Service integration→Buses→CuramBus*;
2. Select the *Destinations* link in the *Destination resources* section;

3. Click *New* to open the “Create new destination” wizard;
4. Select *Queue* as the destination type and click *Next*;
5. Set the following queue attributes:

   Idenifier: SIB_`<QueueName>`

   Leave everything else as the default and click *Next*;
6. Use the *Selected Bus Member* and click *Next*;
7. Click *Finish* to confirm the queue creation.
8. Select the newly added SIB_`<QueueName>` queue now displayed on the list of existing providers. This will open the configuration screen again;
9. Use the following table to set the Exception Destination via the *Specify* radio button and associated text filed;

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

Table A.2 Exception Destination Settings

10. Click *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. Click *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 click *OK* to apply the changes.

Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

Setup the Required Topics

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. Click *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 click *OK* to apply the changes.
5. Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

Setup the Required Queue Activation Specifications

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.

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 clicking *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 click *OK* to add the port.

Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

Setup the Required Topic Activation Specifications

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)

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

Save the changes to the master configuration as described in Section A.2.6, *Save the Master Configuration*.

A.2.14 Configure Historical Log Files

It is possible to configure the maximum number of historical log files maintained by a particular server. To do this

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. Click *OK* to apply the changes;
7. Save the changes to the master configuration.

A.2.15 Post Configuration

Service Integration Bus Database Tables

After setup, it is necessary to manually create database tables required for the *Service Integration Bus*. *WebSphere Application Server* provides a utility to generate the SQL for creating these tables, the *SIB DDL Generator*.

The generator can be run by executing the following command:

```
          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;
- *database_name* is the name of the database to be used.

For example:

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

This command will output some SQL statements and this output should then be executed on the target database.

Timer Service Database Tables

After setup, it is necessary to manually create the database tables required for the *Timer Service*. *WebSphere Application Server* provides the DDL for these tables in it's `WAS_HOME/Scheduler` directory.

The DDL files that should be 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.

A.2.16 Completion

The application server is now configured and ready to install an *IBM Cúram Social Program Management* application on it. Log out of the *Administration Console* and restart the *WebSphere* application server using the targets description in Section 3.5, *Starting and Stopping WebSphere Servers*.

A.3 Manual Application Deployment

To install an enterprise application in *WebSphere Application Server*, the *Administration Console* can be used. The steps below describe how to 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. 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 Section A.2.6, *Save the Master Configuration* 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 Section 4.2.2, *Change SYSTEM Username* 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.

    a. Check *Select* for the mdbuser role and click *Map Users...*;

    b. Enter the appropriate username in the *Search String* field and click *Search*;

    c. Select the ID from the *Available:* list and click >> to add it to the *Selected:* list and click *OK*.

    d. 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.

A.4 WebSphere Network Deployment

IBM's WebSphere Application Server Network Deployment offers advanced deployment services, including clustering, edge services and high availability for distributed configurations. The *Cúram Third Party Tools Installation Guide* should be consulted for more information on the installation of *WebSphere Network Deployment*.

A.4.1 Creating the Profiles

After installing *WebSphere Network Deployment*, it is necessary in most cases to create at least two profiles. One will act as the deployment manager for the node and the others as the federated servers.

This is done via the *Profile Creation Wizard*, which is started via the `pct<hardware platform>` file from the `bin/ProfileCreator` directory of the *WebSphere Application Server* installation.

The first choice of note during 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. It is recommended that administrative security is enabled on profile creation. These settings can be changed later.

A.4.2 Federating a Node

The federation of an application server profile requires the targeted *Deployment Manager* to be started.

The *Deployment Manager* can be started by running the following command 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 the following command is used 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*.

A.4.3 Configuration of Node

Before deploying an application on the registered node, the server must first be configured. This is done through the *Deployment Manager Administration Console* and 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 *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/stopped from the list and click the *Start* or *Stop* button as required.

The next step in the process is to configure the federated servers. As mentioned before, all configuration is done through the *Deployment Manager Administrative Console*. Section A.2, *Manual WebSphere Application Server Configuration* 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. Click the *Save* button. The synchronization may take some time;
4. Check the system and/or *WebSphere Application Server* logs for synchronization completion. These messages may 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 *WebSphere Application Server* logs to ensure success;

Section A.2.7, *Configure Administration Security* 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.

Note

Before building the `Curam.ear` for deployment it is worth noting the *BOOTSTRAP_ADDRESS* of the server that these will be 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*.

A.4.4 Deploying on the Node

Finally, Section A.3, *Manual Application Deployment* 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

Programming Interface Information

This publication documents intended programming interfaces that allow the customer to write programs to obtain the services of IBM Cúram Social Pogram Management.

Trademarks