Oracle® Retail Store Inventory Management

Security Guide

Release 15.0

E68593-01

December 2015

ORACLE®

Oracle Retail Store Inventory Management Security Guide, Release 15.0

E68593-01

Primary Author: Philip Wells

reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

2 Post Installation of Retail Infrastructure in Database

3 Post Installation of Retail Infrastructure in WebLogic

4 Troubleshooting

5 Importing Topology Certificate

6 Using Self Signed Certificates

Send Us Your Comments

Oracle Retail Store Inventory Management Security Guide, Release 15.0.

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

The *Oralce Retail Store Inventory Management Security Guide* serves as a guide for administrators, developers, and system integrators who securely administer, customize, and integrate the Oracle Retail Store Inventory Management application. Installation and configuration for is covered in more detail in the Oracle Retail store Inventory Management Installation Guide.

Audience

This document is intended for administrators, developers, and system integrators who perform the following functions:

- Document specific security features and configuration details for the Oracle Retail MP& O Suite products, in order to facilitate and support the secure operation of the Oracle Retail product and any external compliance standards.
- Guide administrators, developers, and system integrators on secure product implementation, integration, and administration. Functional and technical description of the problem (include business impact).

It is assumed that the readers have general knowledge of administering the underlying technologies and the application.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail Store Inventory Management Release 15.0 documentation set:

- *Oracle Retail Store Inventory Management Release Notes*
- *Oracle Retail Store Inventory Management Operations Guide*

- *Oracle Retail Store Inventory Management Implementation Guide, Volume 1*
- *Oracle Retail Store Inventory Management Implementation Guide, Volume 2*
- *Oracle Retail Store Inventory Management Implementation Guide, Volume 3*
- *Oracle Retail Store Inventory Management Implementation Guide, Volume 4*
- *Oracle Retail Store Inventory Management Implementation Guide, Volume 5*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 15.0) or a later patch release (for example, 15.1). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Oracle Retail Applications

The following chapters provide guidance for administrators, developers, and system integrators who securely administer, customize, and integrate the Oracle Retail Applications.

Part I contains the following chapters:

1

Pre-installation of Retail Infrastructure in WebLogic

Oracle Retail applications are primarily deployed in Oracle WebLogic server as Middleware tier. Java and forms based applications rely upon Middleware infrastructure for complete security apart from application specific security features.

This chapter describes the pre-installation steps for secured setup of Oracle Retail infrastructure in WebLogic.

The following topics are covered in this chapter:

Pre-installation - Steps for Secured Setup of Oracle Retail Infrastructure in WebLogic

Secured Sockets Layer (SSL) protocol allows client-server applications to communicate across a network in a secured channel. Client and server should both decide to use SSL to communicate secured information like user credentials or any other secured information.

WebLogic Server supports SSL on a dedicated listen port. Oracle Forms are configured to use SSL as well. To establish an SSL connection, a Web browser connects to WebLogic Server by supplying the SSL port and the Hypertext Transfer Protocol (HTTPs) protocol in the connection URL.

For example: `https://myserver:7002`

Retail Merchandising System (RMS) setup is supported in WebLogic in secured mode. For enterprise deployment, it is recommended to use SSL certificates signed by certificate authorities.

Note: You need to obtain a separate signed SSL certificates for each host where application is being deployed.

The Security Guide focuses on securing Oracle Retail Applications in single node setup and not on applications deployed on clusters.

Certificate Authority

Certificate Authority or Certification Authority (CA) is an organization which provides digital certificates to entities and acts as trusted third party. Certificates issued by the commercial CAs are automatically trusted by most of the web browsers, devices, and applications. It is recommended to have certificates obtained from a trusted CA or commercial CAs to ensure better security.

Obtaining an SSL Certificate and Setting up a Keystore

Note: SSL certificates are used to contain public keys. With each public key there is an associated private key. It is critically important to protect access to the private key. Otherwise, the SSL messages may be decrypted by anyone intercepting the communications.

Perform the following steps to obtain an SSL certificate and setting up a keystore:

1. Obtain an identity (private key and digital certificates) and trust (certificates of trusted certificate authorities) for WebLogic Server.
2. Use the digital certificates, private keys, and trusted CA certificates provided by the WebLogic Server kit, the CertGen utility, Sun Microsystem's keytool utility, or a reputed vendor such as Entrust or Verisign to perform the following steps:
   a. Set appropriate JAVA_HOME and PATH to java, as shown in the following example:

      ```
      export JAVA_HOME=/u00/webadmin/product/jdk
      export PATH=$JAVA_HOME/bin:$PATH
      ```

   b. Create a new keystore.

      keytool -genkey -keyalg RSA -keysize 2048 -keystore <keystore> -alias <alias>

      For example:

      ```
      keytool -genkey -keyalg RSA -keysize 2048 -keystore hostname.keystore
      -alias hostname
      ```

   c. Generate the signing request.

      keytool -certreq -keyalg RSA -file <certificate request file> -keystore <keystore> -alias <alias>

      For example:

      ```
      keytool -certreq -keyalg RSA -file hostname.csr -keystore hostname.keystore
      -alias hostname
      ```

   d. Submit the certificate request to CA.

3. Store the identity and trust.

   Private keys and trusted CA certificates which specify identity and trust are stored in a keystore.

   In the following examples the same keystore to store all certificates are used:

   a. Import the root certificate into the keystore as shown in the following example:

      ```
      keytool -import -trustcacerts -alias  verisignclass3g3ca -file Primary.pem
      -keystore  hostname.keystore
      ```

      A root certificate is either an unsigned public key certificate or a self-signed certificate that identifies the Root CA.

   b. Import the intermediary certificate (if required) into the keystore as shown in the following example:

      ```
      keytool -import -trustcacerts -alias oracleclass3g3ca -file Secondary.pem
      -keystore  hostname.keystore
      ```

   c. Import the received signed certificate for this request into the keystore as shown in the following example:

      ```
      keytool -import -trustcacerts -alias hostname -file cert.cer -keystore
      hostname.keystore
      ```

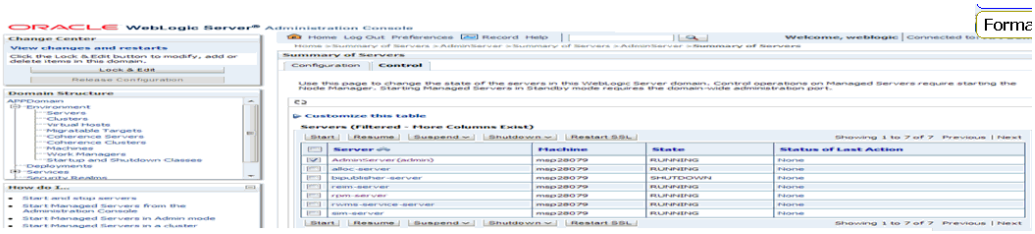
Creating a WebLogic Domain

WebLogic domain is created for Oracle Retail Applications as part of the installation. Different domains are created in different hosts for different applications in situations where applications are being managed by different users or deployed on different hosts. Once the domains are created, you need to enable the SSL ports if not done already.

Perform the following steps to enable the SSL:

1. Log in to WebLogic console using Administrator user. For example, weblogic.
2. Navigate to <Domain> > Environment > Servers > < Servername> > Configuration > General tab.
3. Click **Lock & Edit**.
4. Select **SSL Listen Port Enabled** and assign the port number.
5. Click **Save** and **Activate Changes**.
6. Restart SSL to enable the changes.

Figure 1–1 Restarting the Admin Server



Configuring the Application Server for SSL

Perform the following steps to configure the Application Server for SSL:

1. Configure the identity and trust keystores for WebLogic Server in the WebLogic Server Administration Console.
   a. In the Change Center of the Administration Console, click **Lock & Edit**.
   b. In the left pane of the Console, expand **Environment** and select **Servers**.
   c. Click the name of the server for which you want to configure the identity and trust keystores as shown in the following example:

      ```
      WLS_FORMS is for Forms server
      ```

   d. Select **Configuration**, then select **Keystores**.
   e. In the **Keystores** field, select the method for storing and managing private keys/digital certificate pairs and trusted CA certificates.

      The following options are available:

      - **Demo Identity and Demo Trust** - The demonstration identity and trust keystores, located in the BEA_HOME\server\lib directory and the Java Development Kit (JDK) cacerts keystore, are configured by default. You need to use for development purpose only.

      - **Custom Identity and Java Standard Trust** - A keystore you create and the trusted CAs defined in the cacerts file in the JAVA_HOME\jre\lib\security directory.

      - **Custom Identity and Custom Trust [Recommended]** - An Identity and trust keystores you create.

      - **Custom Identity and Command Line Trust**: An identity keystore you create and command-line arguments that specify the location of the trust keystore.

   f. Select **Custom Identity** and **Custom Trust**.
   g. In the **Identity** section, define the following attributes for the identity keystore:

      - **Custom Identity Keystore** - This is the fully qualified path to the identity keystore.

      - Custom Identity Keystore Type - This is the type of the keystore. Generally, this attribute is Java KeyStore (JKS); if it is left blank, it defaults to JKS.

      - Custom Identity Keystore Passphrase - This is the password you must enter when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore so whether or not you define this property depends on the requirements of the keystore.

   h. In the **Trust** section, define properties for the trust keystore.

      If you choose **Java Standard Trust** as your keystore, specify the password defined when creating the keystore.

   i. Confirm the password.

      If you choose **Custom Trust [Recommended]** define the following attributes:

      - **Custom Trust Keystore** - This is the fully qualified path to the trust keystore.

      - **Custom Trust Keystore Type** - This is the type of the keystore. Generally, this attribute is JKS; if it is left blank, it defaults to JKS.

- **Custom Trust Keystore Passphrase** - This is the password that you need to enter when reading or writing to the keystore. This attribute is optional or required depending on the type of keystore. All keystores require the passphrase in order to write to the keystore. However, some keystores do not require the passphrase to read from the keystore. WebLogic Server only reads from the keystore, so whether or not you define this property depends on the requirements of the keystore.

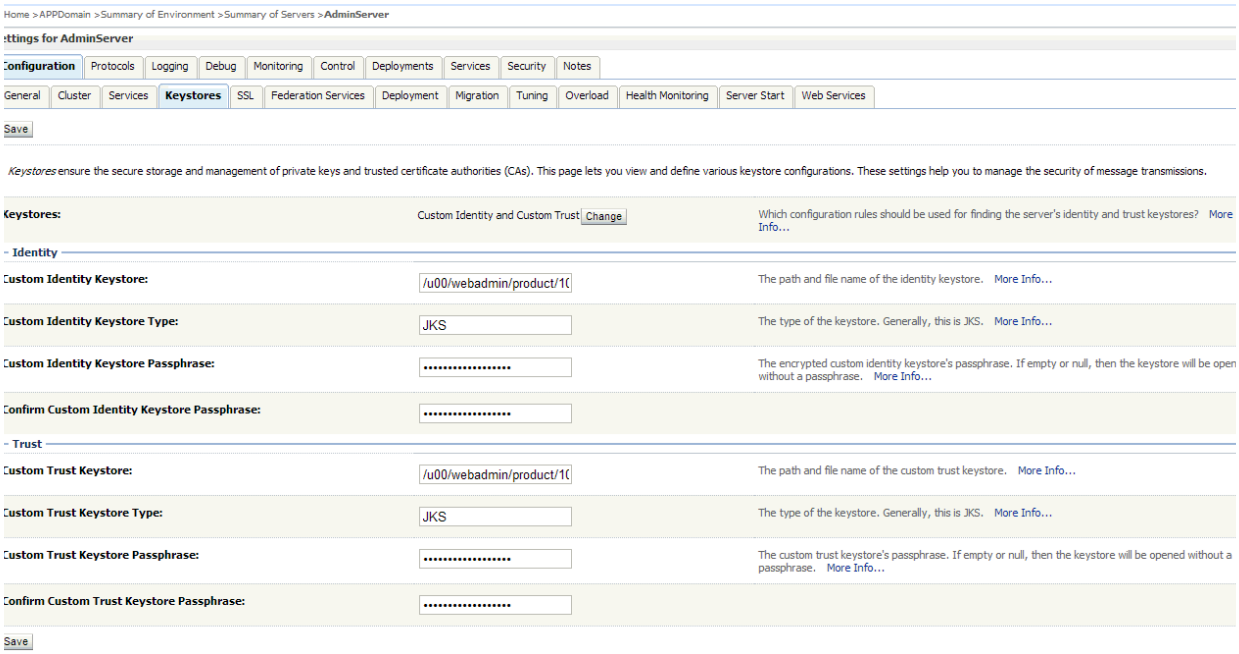
j. Click **Save**.

k. To activate these changes, in the Change Center of the Administration Console, click **Activate Changes**.

Note: Not all changes take effect immediately, some require a restart.

Figure 1–2 shows how to configure the Application Server for SSL.

Figure 1–2 Configuring the Identity and Trust Keystores for WebLogic Server



For more information on configuring Keystores, see the *Administration Console Online Help*.

2. Set SSL configuration options for the private key alias and password in the WebLogic Server Administration Console.

   a. In the Change Center of the Administration Console, click **Lock & Edit**.

   b. In the left pane of the Console, expand **Environment** and select **Servers**.

   c. Click the name of the server for which you want to configure the identity and trust keystores.

   d. Select **Configuration**, then select **SSL**.

   e. In the **Identity and Trust Locations**, the **Keystore** is displayed by default.

f. In the **Private Key Alias**, type the string alias that is used to store and retrieve the server's private key.

g. In the **Private Key Passphrase**, provide the keystore attribute that defines the passphrase used to retrieve the server's private key.

h. Save the changes.

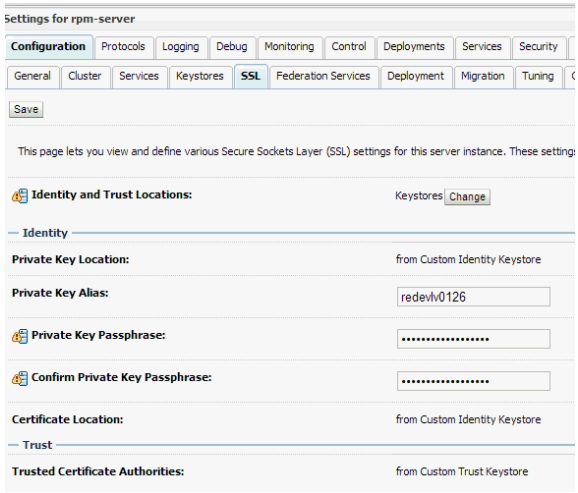
i. Click **Advanced** section of SSL tab.

j. In the **Hostname Verification**, select **None**.

This specifies to ignore the installed implementation of the WebLogic.security.SSL.HostnameVerifier interface (this interface is generally used when this server is acting as a client to another application server).

k. Save the changes.

Figure 1–3 Configuring SSL



For more information on configuring SSL, see the section *Configure SSL* in the *Administration Console Online Help*.

All the server SSL attributes are dynamic; when modified through the Console. They cause the corresponding SSL server or channel SSL server to restart and use the new settings for new connections. Old connections will continue to run with the old configuration. You must reboot WebLogic Server to ensure that all the SSL connections exist according to the specified configuration.

Use the **Restart SSL** button on the **Control**: Start/Stop page to restart the SSL server when changes are made to the keystore files. You have to apply the same for subsequent connections without rebooting WebLogic Server.

Upon restart you can see the following similar entries in the log:

```
<Mar 11, 2013 5:18:27 AM CDT> <Notice> <WebLogicServer> <BEA-000365> <Server
state changed to RESUMING>
<Mar 11, 2013 5:18:27 AM CDT> <Notice> <Server> <BEA-002613> <Channel
"DefaultSecure" is now ing on 10.141.15.214:57002 for protocols iiops, t3s,
ldaps, https.>
<Mar 11, 2013 5:18:27 AM CDT> <Notice> <Server> <BEA-002613> <Channel
"DefaultSecure[1]" is now ing on 127.0.0.1:57002 for protocols iiops, t3s,
ldaps, https.>
<Mar 11, 2013 5:18:27 AM CDT> <Notice> <WebLogicServer> <BEA-000329> <Started
```

```
WebLogic Admin Server "AdminServer" for domain "APPDomain" running in
Production Mode>
<Mar 11, 2013 5:18:27 AM CDT> <Notice> <WebLogicServer> <BEA-000365> <Server
state changed to RUNNING>
<Mar 11, 2013 5:18:27 AM CDT> <Notice> <WebLogicServer> <BEA-000360> <Server
started in RUNNING mode>
```

Note: For complete security of the WebLogic Server, it is recommended to secure both Administration as well the Managed Server where application is being deployed. You can choose to disable the non-SSL ports (HTTP). It is recommended to secure the Node Manager.

The steps to secure Node Manager is provided in the following section.

Configuring WebLogic Scripts if Admin Server is Secured

Perform the following steps to configure the WebLogic scripts if Admin Server is secured:

1. Update the WebLogic startup/shutdown scripts with secured port and protocol to start/stop services.
2. Backup and update the following files in <DOMAIN_HOME>/bin with correct Admin server urls:

   startManagedWebLogic.sh: echo "$1 managedserver1 http://apphost1:7001"

   stopManagedWebLogic.sh: echo "ADMIN_URL defaults to t3://apphost1:7001 if not set as an environment variable or the second command-line parameter."

   stopManagedWebLogic.sh: echo "$1 managedserver1 t3://apphost1:7001 WebLogic

   stopManagedWebLogic.sh: ADMIN_URL="t3://apphost1:7001"

   stopWebLogic.sh: ADMIN_URL="t3://apphost1:7001"
3. Change the URLs as follows:

   t3s://apphost1:7002

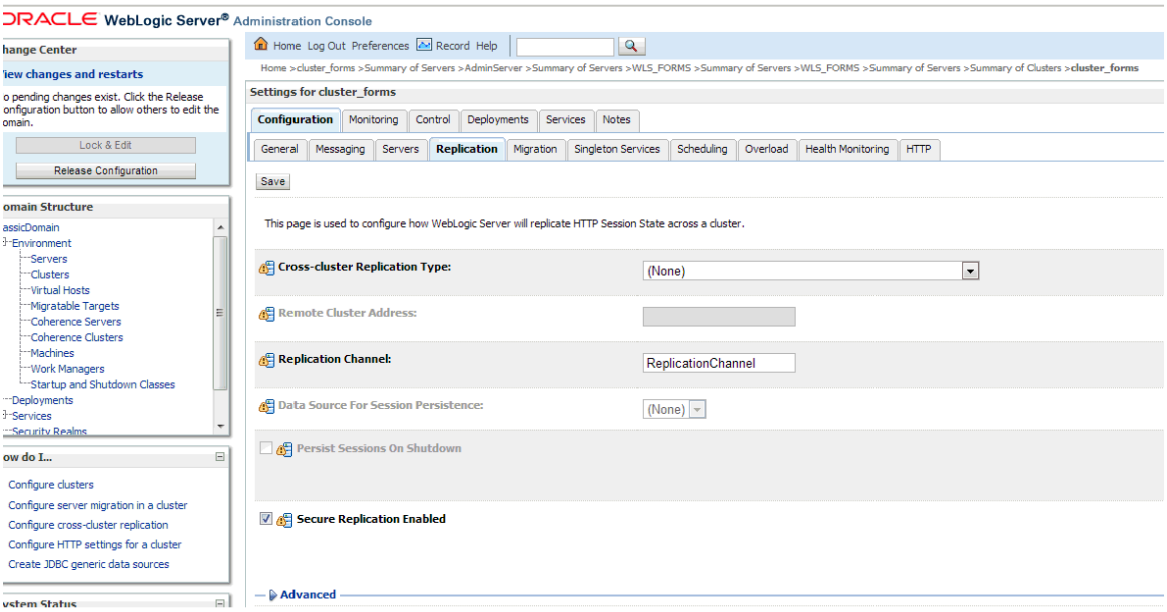
   https://apphost1:7002

Additional Configuration for WLS_FORMS (For forms server)

Perform the following steps for WebLogic forms:

1. Log in to **WebLogic Console**. Select **Environment** > **Clusters** > **cluster_forms**, then select **Configuration** > **Replication**.
2. Select **Secure Replication Enabled**.
3. Start the **WLS_FORMS** Managed server.

Figure 1–4 WebLogic Server Forms



Adding Certificate to the JDK Keystore for Installer

You will need the Oracle Retail Application installer to run Java. In situations where Administration Server is secured using signed certificate, the Java keystore through which the installer is launched must have the certificate installed.

In case the installer is being run using JDK deployed at location /u00/webadmin/product/jdk, follow the steps as shown in Example 1–1.

Example 1–1 Adding certificate to the JDK keystore for Installer

```
apphost1:[10.3.6_apps] /u00/webadmin/ssl> keytool -import -trustcacerts -alias
apphost1 -file /u00/webadmin/ssl/apphost1.cer -keystore
/u00/webadmin/product/jdk/jre/lib/security/cacerts
Enter keystore password:
Certificate was added to keystore
apphost1:[10.3.6_apps] /u00/webadmin/ssl>
```

Enforcing Stronger Encryption in WebLogic

It is recommended to use a stronger encryption protocol in your production environment.

See the following sections to enable the latest SSL and cipher suites.

SSL protocol version configuration

In a production environment, Oracle recommends Transport Layer Security (TLS) Version 1.1, or higher for sending and receiving messages in an SSL connection.

To control the minimum versions of SSL Version 3.0 and TLS Version 1 that are enabled for SSL connections, do the following:

- Set the **webLogic.security.SSL.minimumProtocolVersion=protocol** system property as an option in the command line that starts WebLogic Server.

This system property accepts one of the following values for protocol:

Figure 1–5 Values for Protocol of System Property

Value	Description
SSLv3	Specifies SSL V3.0 as the minimum protocol version enabled in SSL connections.
TLSv1	Specifies TLS V1.0 as the minimum protocol version enabled in SSL connections.
TLSvx.y	Specifies TLS V*x*.*y* as the minimum protocol version enabled in SSL connections, where: • *x* is an integer between 1 and 9, inclusive • *y* is an integer between 0 and 9, inclusive For example, TLSv1.2.

- Set the following property in startup parameters in WebLogic Managed server for enabling the higher protocol:

  Dweblogic.security.SSL.minimumProtocolVersion=TLSv1.1

> **Note:** In case protocol is set for Managed servers, the same should be set for Administration server. Ensure that all the managed servers are down when making changes to the Administration server for setting up the protocol. It is recommended to set the properties in Administration server and then the Managed server.

Upgrading JDK to Use Java Cryptography Extension

You need to install the unlimited encryption Java Cryptography Extension (JCE) policy, if you want to use the strongest Cipher suite (256 bit encryption) AES_256 (**TLS_RSA_WITH_AES_256_CBC_SHA**). It is dependent on the Java Development Kit (JDK) version.

Using the following URL, download and install the JCE Unlimited Strength Jurisdiction Policy Files that correspond to the version of your JDK:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

For JDK 7, download from the following URL:

http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html and replace the files in JDK/jre/lib/security directory

> **Note:** Restart the entire WebLogic instance using the JDK to enable changes to take effect once the JCE has been installed.

Enabling Cipher in WebLogic SSL Configuration

Configure the <ciphersuite> element in the <ssl> element in the <DOMAIN_HOME>\server\config\config.xml file in order to enable the specific Cipher Suite to use as follows:

> **Note:** You need to ensure that the tag <ciphersuite> is added immediately after tab <enabled>.

```
<ssl>
<name>examplesServer</name>
<enabled>true</enabled>
<ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
```

```
<-port>17002</-port>
...
</ssl>
```

Note: The above can be done using wlst script.

For more information, go to http://docs.oracle.com/cd/E24329_01/web.1211/e24422/ssl.htm#BABDAJJG. It is advisable to bring down the managed server prior to making the changes.

Securing Nodemanager with SSL Certificates

Perform the following steps for securing the Nodemanager with SSL certificates:

1. Navigate to **<BEA_HOME>/wlserver_10.3/common/nodemanager** and take a backup of nodemanager.properties.
2. Add the following similar entries to nodemanager.properties:

   KeyStores=CustomIdentityAndCustomTrust

   CustomIdentityKeyStoreFileName=/u00/webadmin/ssl/hostname.keystore

   CustomIdentityKeyStorePassPhrase=[password to keystore, this will get encrypted]

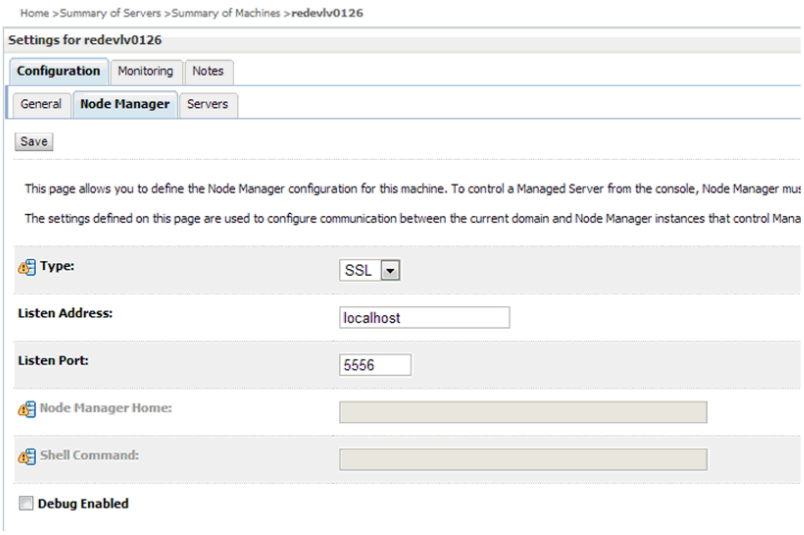
   CustomIdentityAlias=hostname

   CustomIdentityPrivateKeyPassPhrase=[password to keystore, this will get encrypted]

   CustomTrustKeyStoreFileName=/u00/webadmin/ssl/hostname.keystore

   SecureListener=true
3. Log in to **WebLogic console**, navigate to **Environment**, and then **Machines**.
4. Select the nodemanager created already and navigate to **Node Manager** tab.
5. In the Change Center, click **Lock & Edit**.
6. In the **Type** field, select **SSL** from the list.
7. Click **Save** and **Activate**.

Figure 1–6 Securing the Nodemanager



8. You need to bounce the entire WebLogic Domain for changes to take effect, after activating the changes.
9. You need to verify if the nodemanager is reachable in **Monitoring** tab after restart.

Using Secured Lightweight Directory Access Protocol (LDAP)

The Application can communicate with LDAP server on a secured port. It is recommended to use the secured LDAP server to protect user names and passwords from being sent in clear text on the network.

For information on Configuring Secure Sockets Layer (SSL), see the *Oracle Fusion Middleware Administration Guide*.

It is important to import the certificates used in LDAP server into the Java Runtime Environment (JRE) of the WebLogic server for SSL handshake, in case the secure LDAP is used for authentication.

For example:

1. Set JAVA_HOME and PATH to the JDK being used by WebLogic Domain.
2. Backup the JAVA_HOME/jre/lib/security/cacerts

   /u00/webadmin/product/jdk/jre/lib/security> cp -rp cacerts cacerts_ORIG
3. Import the Root and Intermediary (if required) certificates into the java keystore.

   /u00/webadmin/product/jdk/jre/lib/security> keytool -import -trustcacerts -alias verisignclass3g3ca -file ~/ssl/Primary.pem -keystore cacerts

   /u00/webadmin/product/jdk/jre/lib/security> keytool -import -trustcacerts -alias oracleclass3g3ca -file ~/ssl/Secondary.pem -keystore cacerts
4. Import the User certificate from LDAP server into the java keystore.

   /u00/webadmin/product/jdk/jre/lib/security> keytool -import -trustcacerts -alias hostname -file ~/ssl/cert.cer -keystore cacerts

Note: The default password of JDK keystore is **changeit**.

The deployed application should be able to communicate with LDAP on SSL port after successful SSL Handshake.

Connecting from Forms Application to Secured Database

RMS and Oracle Retail Warehouse Management System (RWMS) connect to the database using the Transparent Network Substrate (TNS) Alias as setup in tnsnames.ora file available in the location mentioned in RMS or RWMS environment file created during installation. Example 1–2 refers to an RMS Forms environment file, but the same steps apply to RWMS.

Example 1–2 Identify TNS_ADMIN setting in environment file created during installation

```
$ grep TNS_ADMIN <WLS_HOME> /user_
projects/domains/ClassicDomain/config/fmwconfig/servers/WLS_
FORMS/applications/formsapp_11.1.2/config/develop/rmsFqa3.env
TNS_ADMIN=/u00/webadmin/product/10.3.X_FORMS/WLS/asinst_1/config
```

For secured setup, the TNS Alias inside tnsnames.ora should refer to the TCPS port for Secured Listener of the database.

Example 1–3 Referring TNS Alias inside tnsnames.ora to the TCPS port for Secured Listener of the database

```
<DB_NAME>_secure =
(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = tcps)(host = dbhost1)(Port =
2484)))
(CONNECT_DATA = (SID = <DB_NAME>) (GLOBAL_NAME = <DB_NAME>)))
```

Enabling Access to Secured Database from Forms Oracle Home - Optional

You need to perform the following additional setup to connect to Oracle database on secured port (TCPs) from Forms Oracle Home:

1. Create wallet using orapki.

Note: A wallet is created using either orapki or mkstore utility. Forms installation provides orapki utility to create the wallet and is used for creation of wallet.

```
$ mkdir /u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_FRHome1/network/wallet
$ cd /u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_FRHome1/network/wallet

$  export JAVA_HOME=/u00/webadmin/product/jdk
$  export PATH=$JAVA_HOME/bin:$PATH

$  export ORACLE_HOME=/u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_FRHome1
$  export PATH=$ORACLE_HOME/bin:$PATH
$  export PATH=/u00/webadmin/product/10.3.X_FORMS/WLS/oracle_common/bin:$PATH
$  orapki wallet create -wallet
/u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_FRHome1/network/wallet/secured
-auto_login -pwd <wallet-pwd>
Oracle PKI Tool: Version 11.1.1.5.0
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
$  ls
```

```
    cwallet.sso  ewallet.p12
```

2. Import the Signed certificates into the wallet.

Example 1–4 Importing all certificates into the wallet

```
$ orapki wallet jks_to_pkcs12 -wallet
/u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_FRHome1/network/wallet/secured -pwd
<wallet-pwd> -keystore
/u00/webadmin/product/10.3.X_APPS/WLS/wlserver_10.3/server/lib/apphost1.keystore
-jkspwd <keystore-pwd>
Oracle PKI Tool: Version 11.1.1.5.0
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
```

For information on Oracle Wallet Manager and orapki, see *Fusion Middleware Administrator's Guide*.

3. Provide the wallet details in sqlnet.ora file.

> **Note:** You need to create a sqlnet.ora file with details of the wallet in $ORACLE_HOME/network/admin directory, if the file is not available.

Example 1–5 sqlnet.ora file

```
SQLNET.AUTHENTICATION_SERVICES=(TCPS,NTS)
SSL_CLIENT_AUTHENTICATION = TRUE
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA = (DIRECTORY = /u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_
FRHome1/network/wallet/secured))
)
```

4. Connect using sqlplus.

```
$ export ORACLE_HOME=/u00/webadmin/product/10.3.X_FORMS/WLS/Oracle_FRHome1
$ export PATH=$ORACLE_HOME/bin:$PATH

$ sqlplus rms01app@<DB_NAME>_secure
SQL*Plus: Release 12.1.0.1.0 Production on Tue Aug 5 02:15:22 2014
Copyright (c) 1982, 2013, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

Webservice Security Policies

You need to configure the user credentials and other security related information at the service consumer and the app service provider layers, in order to provide end to end security between Web service consumer and provider.

The security policies certified by Oracle Retail are as follows:

1. Username Token over HTTPS - This security configuration is referred as Policy A in this document. This policy provides confidentiality due to the use of SSL, however it does not provide non-repudiation as nothing is signed.

   Wssp1.2-2007-Https-UsernameToken-Plain.xml

2. Message Protection - This security configuration is referred as Policy B in this document. This policy encrypts the messages itself, so SSL is not used. The sender also signs the messages, which provides non-repudiation of the messages. However, this policy is more complex to implement.
   - Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128
   - Wssp1.2-2007-EncryptBody
   - Wssp1.2-2007-SignBody

   Note:

   - The web services are secured using WebLogic policies (as opposed to OWSM policies).
   - If the application services are secured with any policy other than what is mentioned in this document or custom policies, the instructions in the document will not work.
   - The security setup in the document does not address authorization. Authorization must be taken care by the individual application hosting the services.
   - Policy B is not supported over HTTPS. So ensure that non SSL ports are enabled prior to applying Policy B.

Additional Pre-requisite for Oracle Retail Service Backbone (RSB) Security Policies

Perform the additional pre-requisites for Oracle Retail Service Backbone (RSB) security policies:

1. Create DB schema for OSB [PolicyA][PolicyB].
2. Ensure that <RSB_MDS> schema is created while running Repository Creation Utility (RCU) at <rcuHome>/bin/rcu.
3. Extend RSB Domain with OWSM Extention [PolicyA][PolicyB].
4. Ensure that OSB OWSM Extension-11.1.1.6 is selected, when RSBDomain is being created.

Advanced Infrastructure Security

Depending upon your security need for your production environment, infrastructure where Oracle Retail applications are deployed can be secured.

Ensure the following to secure complete protection of environment:

- Securing the WebLogic Server Host
- Securing Network Connections
- Securing your Database
- Securing the WebLogic Security Service
- Securing Applications

For more information on Ensuring the Security of Your Production Environment, see *Securing a Production Environment for Oracle WebLogic Server 12 C Release 1 (10.3.6) Guide*.

2

Post Installation of Retail Infrastructure in Database

Oracle Retail applications use the Oracle database as the backend data store for applications. In order to ensure complete environment security the database should be secured.

This chapter describes the post installation steps for secured setup of Retail infrastructure in the Database.

The following topics are covered in this chapter:

Configuring SSL Connections for Database Communications

Secure Sockets Layer (SSL) is the standard protocol for secure communications, providing mechanisms for data integrity and encryption. This can protect the messages sent and received by the database to applications or other clients, supporting secure authentication and messaging. Configuring SSL for databases requires configuration on both the server and clients, which include application servers.

This section covers the steps for securing Oracle Retail Application Clusters (RAC) database. Similar steps can be followed for single node installations also.

Configuring SSL on the Database Server

The following steps are one way to configure SSL communications on the database server:

1. Obtain an identity (private key and digital certificate) and trust (certificates of trusted certificate authorities) for the database server from a Certificate Authority.
2. Create a folder containing the wallet for storing the certificate information. For Real Application Cluster (RAC) systems, this directory can be shared by all nodes in the cluster for easier maintenance.

   mkdir -p /oracle/secure_wallet

3. Create a wallet in the path. For example,

```
orapki wallet create -wallet /oracle/secure_wallet -auto_login
```

4. Import each trust chain certificate into the wallet as shown in the following example:

```
orapki wallet add -wallet /oracle/secure_wallet -trusted_cert -cert <trust
chain certificate>
```

5. Import the user certificate into the wallet, as shown in the following example:

```
orapki wallet add -wallet /oracle/secure_wallet -user_cert -cert <certificate
file location>
```

6. Update the listener.ora by adding a TCPS protocol end-point first in the list of end points

```
LISTENER1=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=<dbserver>)(PORT=2484))
     (ADDRESS=(PROTOCOL=tcp)(HOST=<dbserver>)(PORT=1521)))
```

7. Update the listener.ora by adding the wallet location and disabling SSL authentication.

```
WALLET_LOCATION =
 (SOURCE=
  (METHOD=File)
  (METHOD_DATA=
   (DIRECTORY=wallet_location)))
SSL_CLIENT_AUTHENTICATION=FALSE
```

8. Update the sqlnet.ora with the same wallet location information and disabling SSL authentication.

```
WALLET_LOCATION =
 (SOURCE=
  (METHOD=File)
  (METHOD_DATA=
   (DIRECTORY=wallet_location)))
SSL_CLIENT_AUTHENTICATION=FALSE
```

9. Update the tnsnames.ora to configure a database alias using TCPS protocol for connections.

```
<dbname>_secure=
  (DESCRIPTION=
   (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCPS)(HOST=<dbserver>)(PORT=2484)))
   (CONNECT_DATA=(SERVICE_NAME=<dbname>)))
```

10. Restart the database listener to pick up listener.ora changes.
11. Verify the connections are successful to the new <dbname>_secure alias
12. At this point either the new secure alias can be used to connect to the database, or the regular alias can be modified to use TCPS protocol.
13. Export the identity certificate so that it can be imported on the client systems

    orapki wallet export -wallet /oracle/secure_wallet -dn <full dn of identity certificate> -cert <filename_to_create>

Configuring SSL on an Oracle Database Client

The following steps are one way to configure SSL communications on the database client:

1. Create a folder containing the wallet for storing the certificate information.

mkdir -p /oracle/secure_wallet

2. Create a wallet in the path. For example,

   orapki wallet create -wallet /oracle/secure_wallet -auto_login

3. Import each trust chain certificate into the wallet as shown in the following example:

   orapki wallet add -wallet /oracle/secure_wallet -trusted_cert -cert <trust chain certificate>

4. Import the identity certificate into the wallet, as shown in the following example:

   orapki wallet add -wallet /oracle/secure_wallet -trusted_cert -cert <certificate file location>

   Note: On the client the identity certificate is imported as a trusted certificate, whereas on the server it is imported as a user certificate.

5. Update the sqlnet.ora with the wallet location information and disabling SSL authentication.

   WALLET_LOCATION =

   (SOURCE=

   (METHOD=File)

   (METHOD_DATA=

   (DIRECTORY=wallet_location)))

   SSL_CLIENT_AUTHENTICATION=FALSE

6. Update the tnsnames.ora to configure a database alias using TCPS protocol for connections.

   <dbname>_secure=

   (DESCRIPTION=

   (ADDRESS_LIST=

   (ADDRESS=(PROTOCOL=TCPS)(HOST=<dbserver>)(PORT=2484)))

   (CONNECT_DATA=(SERVICE_NAME=<dbname>)))

7. Verify the connections are successful to the new <dbname>_secure alias.

8. At this point either the new secure alias can be used to connect to the database, or the regular alias can be modified to use TCPS protocol.

Configuring SSL on a Java Database Connectivity (JDBC) Thin Client

The following steps are one way to configure SSL communications for a Java Database Connectivity (JDBC) thin client:

1. Create a folder containing the keystore with the certificate information.

   mkdir -p /oracle/secure_jdbc

2. Create a keystore in the path. For example,

   ```
   keytool -genkey -alias jdbcwallet -keyalg RSA -keystore /oracle/secure_
   jdbc/truststore.jks -keysize 2048
   ```

3. Import the database certificate into the trust store as shown in the following example:

```
keytool -import -alias db_cert -keystore /oracle/secure_jdbc/truststore.jks
-file <db certificate file>
```

4. JDBC clients can use the following URL format for JDBC connections:

```
jdbc:oracle:thin:@(DESCRIPTION= (ADDRESS= (PROTOCOL=tcps) (HOST=<dbserver>)
(PORT=2484)) (CONNECT_DATA= (SERVICE_NAME=<dbname>)))
```

> **Note:** The <dbname> would be replaced with service name in case of multitenant database (12c).

5. You need to set the properties as shown in Table 2–1, either as system properties or as JDBC connection properties.

Table 2–1 Setting the Properties

Property	Value
javax.net.ssl.trustStore	Path and file name of trust store. For example, /oracle/secure_jdbc/truststore.jks
javax.net.ssl.trustStoreType	JKS
javax.net.ssl.trustStorePassword	Password for trust store

Configuring the Password Stores for Database User Accounts

Wallets can be used to protect sensitive information, including usernames and passwords for database connections. The Oracle Database client libraries have built-in support for retrieving credential information when connecting to databases. Oracle Retail applications utilize this functionality for non-interactive jobs such as batch programs so that they are able to connect to the database without exposing user and password information to other users on the same system.

For information on configuring wallets for database access, see the Appendix Setting Up Password Stores with Oracle Wallet in the product installation guide.

Configuring the Database Password Policies

Oracle Database includes robust functionality to enforce policies related to passwords such as minimum length, complexity, when it expires, number of invalid attempts, and so on. Oracle Retail recommends these policies are used to strengthen passwords and lock out accounts after failed attempts.

For example, to modify the default user profile to lock accounts after five failed login attempts, run the following commands as a database administrator:

1. Query the current settings of the default profile

   select resource_name,limit,resource_type from dba_profiles where profile='DEFAULT';

2. Alter the profile, if failed_login_attempts is set to unlimited:

   alter profile default limit FAILED_LOGIN_ATTEMPTS 5;

Note: Many other profile settings are available for increased security. For more information, see the *Oracle Database Security Guide.*

Configuring SSL for Oracle Data Integrator (ODI)

This section covers the steps for securing Oracle data Integrator (ODI) and communication over HTTPS Protocol.

See the following steps to configure SSL communications for ODI:

1. Set the environment variable for JAVA_HOME as follows:

   $ export JAVA_HOME=/oracle/oracle_linux/jdk1.7 64bit

   $ export PATH=$JAVA_HOME/bin:$PATH

2. Create a Self Signed Keystore. Run the command to generate the keystore as shown in the following example:

```
$ keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks
-storepass password -validity 360 -keysize 2048
What is your first and last name?
  [Unknown]:  <Hostname>
What is the name of your organizational unit?
  [Unknown]:  <Organization Unit>
What is the name of your organization?
  [Unknown]:  <SomeORG>
What is the name of your City or Locality?
  [Unknown]:  <IN>
What is the name of your State or Province?
  [Unknown]:  <MSP>
What is the two-letter country code for this unit?
  [Unknown]:  <US>
Is CN= <Hostname>, OU=<Organization Unit>, O=<SomeORG>, L=<IN>, ST=<MSP>,
C=<US> correct?
  [no]:  yes
Enter key password for <selfsigned>
        (RETURN if same as keystore password):
Re-enter new password:
```

3. Export the certificate from the keystore created above into the file, server.cer:

```
keytool -export -alias selfsigned -storepass password -file server.cer
-keystore keystore.jks
```

   For example:

```
$ keytool -export -alias selfsigned -storepass password -file server.cer
-keystore keystore.jks
Certificate stored in file <server.cer>
```

4. Create the trust-store file, cacerts.jks, and add the server certificate to the trust-store. For example,

```
$ keytool -import -v -trustcacerts -alias selfsigned -file server.cer -keystore
cacerts.jks -keypass password -storepass password
For Example -
$ keytool -import -v -trustcacerts -alias selfsigned -file server.cer -keystore
cacerts.jks -keypass password -storepass password
Owner: CN=<Hostname>, OU=<Organization Unit>, O=<SomeORG>, L=<IN>, ST=<MSP>,
C=<US>
Issuer: CN=<Hostname>, OU=<Organization Unit>, O=<SomeORG>, L=<IN>, ST=<MSP>,
C=<US>
Serial number: 1f5717fd
```

```
Valid from: Fri Aug 01 02:12:50 CDT 2014 until: Mon Jul 27 02:12:50 CDT 2015
Certificate fingerprints:
         MD5:  6E:67:FE:FA:4F:6C:E7:E8:C5:5F:17:97:18:E6:62:7E
         SHA1: 48:B7:66:58:24:C9:BD:A9:F9:E1:FB:08:70:94:35:9A:B0:44:DF:D6
         SHA256:
6A:88:40:E1:A7:2F:67:13:6A:F7:12:D0:F1:47:6C:D7:E8:68:45:73:C3:04:36:24:8A:41:1
8:3D:22:8A:DD:5F
         Signature algorithm name: SHA256withRSA
         Version: 3
Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 2E A6 E1 80 12 33 70 4C   72 FA DF ED 98 BB 33 DF  .....3pLr.....3.
0010: 81 6B 40 A4                                        .k@.
]
]
Trust this certificate? [no]:  yes
Certificate was added to keystore
[Storing cacerts.jks]
```

5. Encode the password used by Keystore and Truststore as follows:

```
$ cd $ODI_HOME/oracledi/agent/bin
$ ./encode.sh <password>
For Example -
$ ./encode.sh password
       fDyp8qdXcuuYUbBcg0Jr
```

6. After configuring repository connection information of the odiparams.sh file, configure and modify the following sections of the file and save it:

```
$ vi odiparams.sh
```

7. Modify the following entries:

```
ODI_KEYSTORE_ENCODED_PASS=fDyp8qdXcuuYUbBcg0Jr
ODI_KEY_ENCODED_PASS=fDyp8qdXcuuYUbBcg0Jr
ODI_TRUST_STORE_ENCODED_PASS=fDyp8qdXcuuYUbBcg0Jr
ODI_JAVA_OPTIONS="-Djava.security.policy=server.policy
-Doracle.security.jps.config=./jps-config.xml
-Djavax.net.ssl.keyStore=<KEYSTORE_LOCATION>/keystore.jks
-Djavax.net.ssl.trustStore=<KEYSTORE_LOCATION>/SSL/cacerts.jks $ODI_PARAMS_
JAVA_OPTION"
```

> **Note:** The encoded password is the one that you generated by running./encode.sh script.

8. Add the following lines to odi.conf before SetJavaHome Environment Variable in the file and save. This will set up ODI Studio for HTTPS.

```
$cd $ODI_HOME/oracledi/agent/bin
Append the odi.conf file with below entries:-
AddVMOption -Djavax.net.ssl.trustStore=<KEYSTORE_LOCATION>/cacerts.jks
AddVMOption -Djavax.net.ssl.trustStorePassword=password
For example
$ vi odi.conf
#Keystore Details
AddVMOption
-Djavax.net.ssl.trustStore=/u03/odi/product/11.1.1.7/SSL/cacerts.jks
AddVMOption -Djavax.net.ssl.trustStorePassword=password
```

Note: The password is the actual password used while encoding. This is not the encoded password.

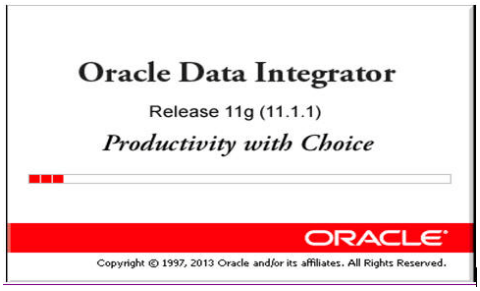
9. Configure ODI STUDIO with New Agent.
10. Go to the following location:

```
$ cd $ODI_HOME/oracledi/client
```

11. Run the following:

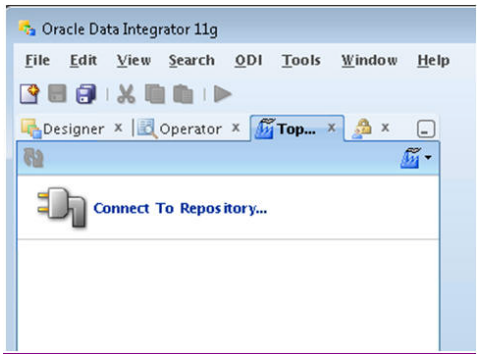
```
$ ./odi.sh
Oracle Data Integrator 12c
Copyright (c) 1997, 2012, Oracle and/or its affiliates. All rights reserved
```

Figure 2–1 Oracle Data Integrator



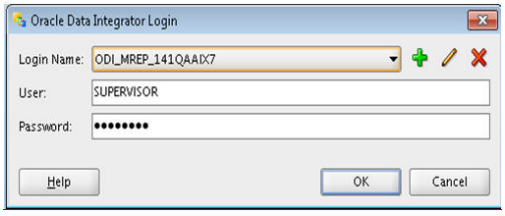
12. The Connect to Repository Window appears:

Figure 2–2 Connect to Repository



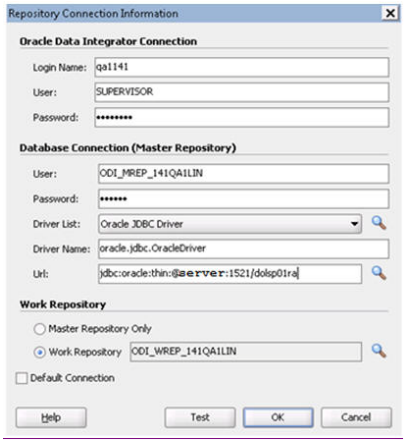
13. Once Connect to Repository Window appears, click **Connect to Repository**. The Oracle Data Integrator Login screen appears.

Figure 2–3 Oracle Data Integrator Login Window



14. Click + symbol and provide the Repository Connection Information.

Figure 2–4 Repository Connection Information Window

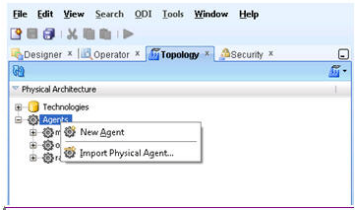


Note: The URL in Figure 2–4 is an example for Pluggable Database. If it is a non-container database, specify the following URL:

jdbc:oracle:thin:@server:1521/<dbname>

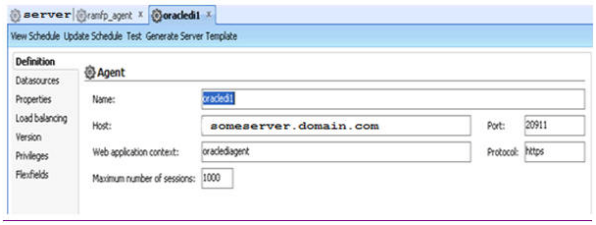
15. Configure the New Agent by right clicking Agents and select New Agent.

Figure 2–5 Creation of ODI Agent



16. Specify all the details and make sure you are using an HTTPS protocol.

Figure 2–6 ODI Agent Connection Information



17. Finally start the ODI Agent to listen on HTTPS port:

```
$ $ODI_HOME/oracledi/agent/bin> ./agent.sh "-PROTOCOL=HTTPS" "-PORT=20911"
"-NAME=oracledi1"
Enter password for TrustStore:
2014-08-01 03:18:45.854 NOTIFICATION ODI-1128 Agent oracledi1 is starting.
Container: STANDALONE. Agent Version: 11.1.1.7.0 - 02/03/2013. Port: 20911. JMX
```

```
Port: 21911.
2014-08-01 03:18:51.209 NOTIFICATION ODI-1111 Agent oracledi1 started. Agent
version: 11.1.1.7.0 - 02/03/2013. Port: 20911. JMX Port: 21911.
2014-08-01 03:18:51.210 NOTIFICATION ODI-1136 Starting Schedulers on Agent
oracledi1.
2014-08-01 03:18:52.040 NOTIFICATION ODI-1137 Scheduler started for work
repository ODI_WREP_141QA1LIN on Agent oracledi1
```

Creating an Encrypted Tablespace in Oracle 12C Container Database

The retail tablespaces can be encrypted in container databases using the following method:

1. Update the SQLNET.ORA file with the following encryption details:
   a. Configure the sqlnet.ora File for a Software Keystore Location.

      ENCRYPTION_WALLET_LOCATION=

      (SOURCE=

      (METHOD=FILE)

      (METHOD_DATA=

      (DIRECTORY=path_to_keystore)))

   b. Restart the listener.
2. Set up the Tablespace Encryption in the Container Database.
   a. Create Software Keystores as follows:

      SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/u03/wallet_cdb' IDENTIFIED BY "val1ue#";

      Keystore altered.

   b. Create an Auto-Login Software Keystore as follows:

      SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE

      '/u03/wallet_cdb' identified by "val1ue#'; Keystore altered.

      Note: The auto-login software keystore can be opened from different computers from the computer where this keystore resides. However, the [local] auto-login software keystore can only be opened from the computer on which it was created.

   c. Open the Software Keystore as follows:

      SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "val1ue#" Container=ALL;

      Keystore altered.

   d. Set the Software TDE Master Encryption Key as follows:

      SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "val1ue#" WITH BACKUP USING 'TDE_ENCRYPTION' Container=all;

      Keystore altered.

Note: One can set the Encryption KEY only for particular PDB if required, by specifying the CONTAINER=<PDB>.

e. Create the ENCRYPTED TABLESPACE in PDB as follows:

SQL> conn sys/D0ccafe1@QOLRP01APP as sysdba

Connected.

SQL> create tablespace test datafile '+DATA1' size 100m ENCRYPTION DEFAULT STORAGE (ENCRYPT);

Tablespace created.

f. Verify the Encryption:

SQL> select * from v$encryption_wallet

WRL_ TYPE	WRL_ PARAMETER	STATUS	WALLET_ TYPE	WALLET OR	FULLY BAC	CON ID
FILE	/u03/wallet_cdb	OPEN	PASSWORD	SINGLE	NO	0

3. For more information on Configuring Transparent Data Encryption (TDE), see

https://docs.oracle.com/cloud/latest/db121/ASOAG/asotrans_config.htm#ASOAG10474

Information may be useful during maintenance activity.

4. Close the Encryption Wallet as follows:

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE Close IDENTIFIED BY "val1ue#" Container=ALL;

Additional Information

For more information on the subjects covered in this section as well as information on other options that are available to strengthen database security, see the *Oracle Database Security Guide 12c Release 1*.

The Oracle Advanced Security Option provides industry standards-based solutions to solve enterprise computing security problems, including data encryption and strong authentication. Some of the capabilities discussed in this guide require licensing the Advanced Security Option.

For more information, see the *Oracle Database Advanced Security Administrator's Guide 12c Release 1*.

3

Post Installation of Retail Infrastructure in WebLogic

This chapter describes the post installation steps for secured setup of Oracle Retail infrastructure in WebLogic.

The following topics are covered in this chapter:

Retail Application Specific Post installation Steps for Security

See the following sections for steps to improve security after an Oracle Retail Application has been installed.

Batch Set Up for SSL Communication

Java batch programs communicate with Java applications deployed in WebLogic. For example, Oracle Retail Price Management (RPM) and Oracle Store Inventory Management (SIM). The communication needs to have SSL handshake with the deployed application. You need to import the SSL Certificates into the JAVA_HOME/jdk/jre/lib/security/cacerts keystore for successful running of the application batches.

Example 3–1 Importing certificates into JDK keystore

```
/u00/webadmin/product/jdk/jre/lib/security> cp -rp cacerts cacerts_ORIG

/u00/webadmin/product/jdk/jre/lib/security> keytool -import -trustcacerts -alias
verisignclass3g3ca -file ~/ssl/Primary.pem -keystore cacerts

/u00/webadmin/product/jdk/jre/lib/security> keytool -import -trustcacerts -alias
oracleclass3g3ca -file ~/ssl/Secondary.pem -keystore cacerts

/u00/webadmin/product/jdk/jre/lib/security> keytool -import -trustcacerts -alias
hostname -file ~/ssl/cert.cer -keystore cacerts
```

Note: The default password of JDK keystore is **changeit**.

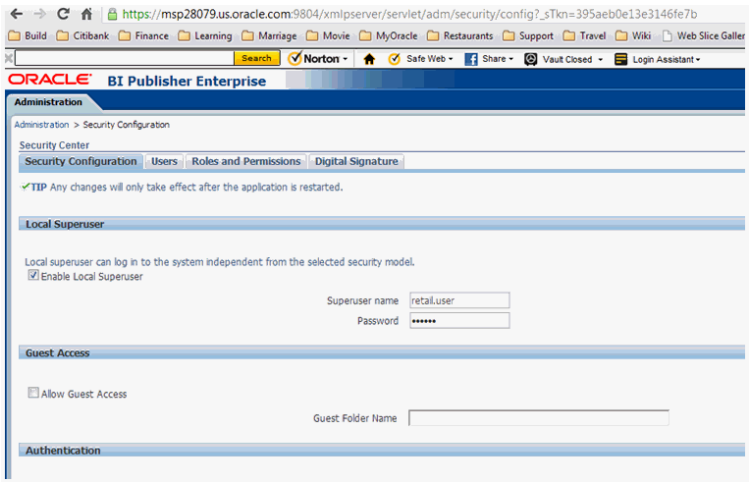
Oracle Business Intelligence (BI) Publisher - Disable Guest User - Optional

The guest account in Oracle Business Intelligence (BI) publisher is used for public facing reports that anyone can see. Disabling this account forces all users to supply their credentials before accessing any information. Disabling guest user enhances security of BI Publisher. However, application which requires guest user will have reporting feature which may cease to function after making this change. For example, RMS reports.

Perform the following steps to disable the guest user:

1. Log in to BI Publisher with user having Administrator privileges.
2. Navigate to Administration > Security Configuration.
3. Deselect **Allow Guest Access**.

Figure 3–1 Administration Window



4. Save and restart the BI Publisher instance.

Retail Merchandising System (RMS) - Forms Timeout Setting - Optional

Oracle Forms can be configured to timeout based on user idle time.

You need to set the following parameters:

1. FORMS_TIMEOUT - This parameter is set RMS/RWMS env file created at <DOMAIN_HOME>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.2/config directory

   The default value for forms timeout is 15 and Valid Values range from 3 to 1440 (1 day).

   This parameter specifies the amount of time in elapsed minutes before the Form Services process is terminated when there is no client communication with the Form Services. Client communication can come from the user doing some work, or from the Forms Client heartbeat if the user is not actively using the form.

2. HeartBeat - This parameter is set in formsweb.cfg file located in DOMAIN_HOME>/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_11.1.1/config directory.

   The default value for **HeartBeat** is 2 and Valid Values range from 1 to 1440 (1 day).

Example:

```
[rmsFqa3]
envfile=./develop/rmsFqa3.env
width=950
height=685
separateFrame=true
form=rtkstrt.fmx
lookAndFeel=Oracle
colorScheme=swan
archive=frmall.jar,icons.jar
imageBase=codebase
heartbeat=12
```

Note: For more information on the above parameters and additional options, see the Oracle Support Note: *Description List For Parameters Affect Timeout In Webforms [ID 549735.1].*

Asynchronous Task JMS Queue Security

This section describes the steps for adding security to the asynchronous task JMS queue. Securing the queue will allow only recognized users of the Retail Application to publish tasks to the JMS queue.

Verifying and Creating Required Async Task Job Role and User

Securing the JMS async task queue requires a special enterprise role and a special user to exist in the retailer's Oracle Internet Directory (OID) instance.

The RETAIL_ASYNC_TASK_JOB is an enterprise role that will be used to group users who will have access to the asynchronous task queue.

The RETAIL _ASYNC_TASK_USER is a special user Retail Applications can use as a principal for executing their message-driven-bean-based consumer processes. This user is a member of the RETAIL_ASYNC_TASK_JOB.

The RETAIL_ASYNC_TASK_JOB and RETAIL_ASYNC_TASK_USER are included as part of the Retail Default Security Reference Implementation installed as part of the Retail Application.

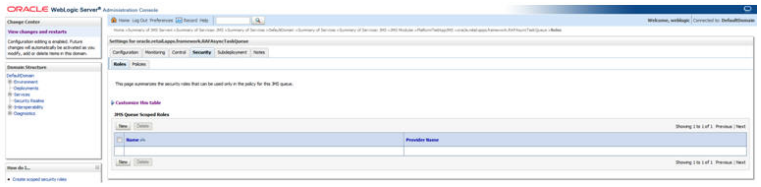
Verify the existence of the job and user in the OID instance. You need to create them if they do not exist.

Securing the Asynchronous Task JMS Queue

Securing the queue can be done through the Weblogic Administration Console by adding a JMS Queue Scoped role.

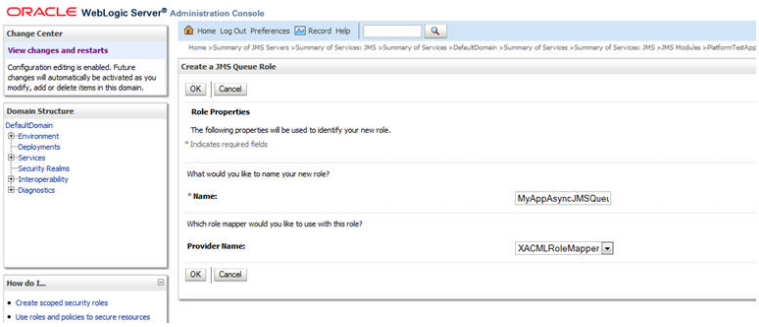
1. Log into the WebLogic Administration Console.
2. Navigate to the JMS Module where the asynchronous task queue belongs to and go to the module's Security tab.
3. Under the Roles section, add a new JMS Queue Scoped Roles.

Figure 3–2 Adding a new JMS Queue Scoped Roles



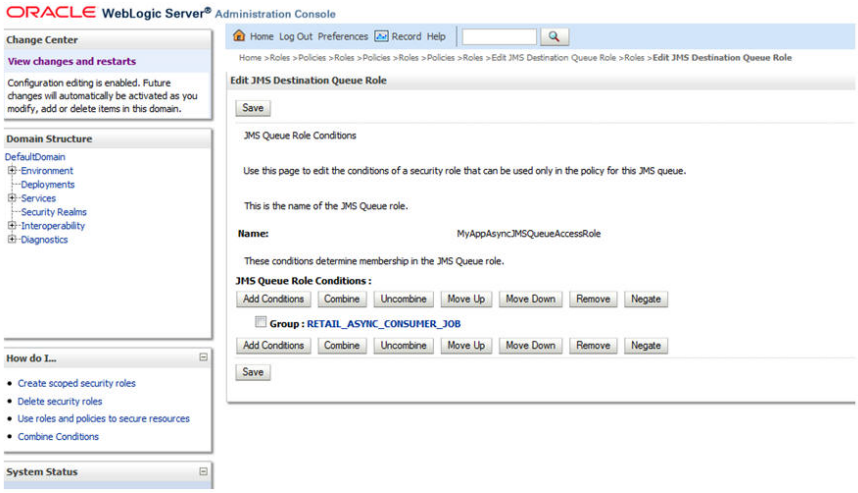
4. Specify a name for the JMS Queue Scoped Role. The suggested naming convention is [AppCode]AsyncJMSQueueAccessRole]. For example, AllocAsyncJMSQueueAccessRole. The JMS Queue Scoped Role will be created.

Figure 3–3 JMS Queue Scoped Role



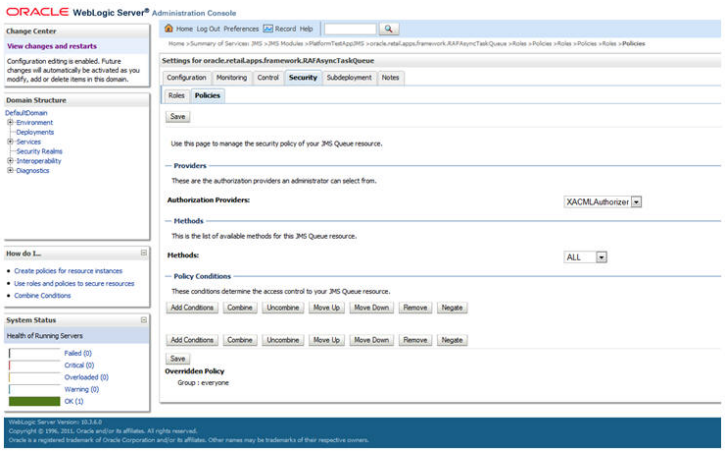
5. Navigate back to the JMS Module's Security tab.
6. Click the JMS Queue Scoped role that was created and add a Group condition for RETAIL_ASYNC_TASK_JOB.

Figure 3–4 Adding a Group condition for RETAIL_ASYNC_TASK_JOB



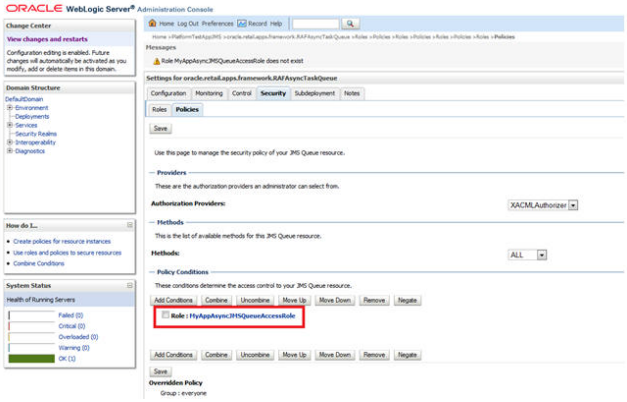
7. Navigate back to the JMS Module's Security tab.
8. Go to the Policies section.

Figure 3–5 Policies Tab



9. Add a new Role based condition specifying the JMS Queue Role created in the previous step.

Figure 3–6 Adding a New Role



10. Save the changes. The queue is now secured.

11. Proceed to the next section to allow the Retail Web Application to publish tasks to the queue.

Allowing Publishing to a Secured Asynchronous Task JMS Queue

Once the Asynchronous Task Queue has been secured with a JMS Queue Scoped Role as described in the previous section, further configuration is required to allow users of the Retail web application to publish tasks to the queue.

The JMS Queue Scoped Role was created to include an enterprise role, RETAIL_ASYNC_TASK_JOB. Any users belonging to this enterprise role will be given access to publish tasks to the queue.

Instead of assigning users directly to the RETAIL_ASYNC_TASK_JOB, it is recommended that applications should identify specific enterprise job roles in their system whose users will be allowed to perform asynchronous processing. Those job roles should be configured to extend from the RETAIL_ASYNC_TASK_JOB group.

See the Oracle Internet Directory documentation for details on how to extend one group to another.

4

Troubleshooting

This chapter covers the common errors, issues, and troubleshooting them.

The following topics are covered in this chapter:

Java Version 7 SSL Handshake Issue while Using Self Signed Certificates

Java Version 7 may have issues using self signed certificates. The self-signed root certificate may not be recognized by Java Version 1.7 and a certificate validation exception might be thrown during the SSL handshake. You need to create the private key with Subject Key Identifier to fix this problem. You need to include an option -addext_ski when the orapki utility is used to create the private key in the root wallet.

Importing the Root Certificate in Local Client JRE

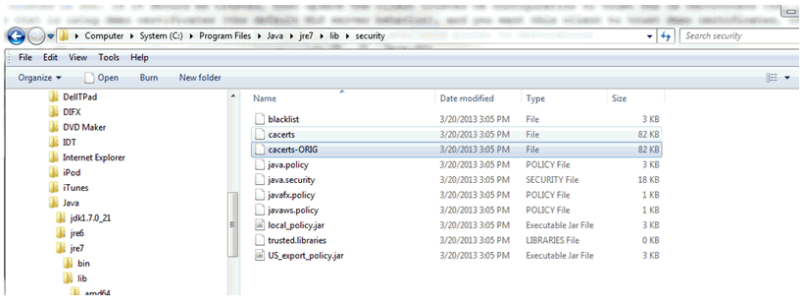
If customers are using certificates other than provided by standard certificate authorities like custom CA implementation, then the JRE used for launching the applications from local machines like laptops or desktops might display a different error messages.

The most probable cause of this issue could be unavailability of root certificates of the CA within the local JRE being used.

Perform the following steps to import the root certificates:

1. Backup cacert at <JRE_HOME>/lib/security/cacert.

Figure 4–1 Cacert Backup



2. Import the certificate using keytool utility as shown in the following example:

```
C:\Program Files\Java\jre7\lib\security>..\..\bin\keytool.exe -import
-trustcacerts -file D:\ADMINISTRATION\SSL\apphost2\Selfsigned\apphost2.root.cer
-alias apphost2 -keystore "C:\Program Files\Java\jre7\lib\security\cacerts"

Enter keystore password: [default is changeit]
Owner: CN=apphost2, OU=<department>, O=<company>,L=<city>,ST=<state or
province>, C=<country>",
Issuer: CN=apphost2, OU=<department>, O=<company>,L=<city>,ST=<state or
province>, C=<country>"
Serial number: 515d4bfb
Valid from: Thu Apr 04 15:16:35 IST 2013 until: Fri Apr 04 15:16:35 IST 2014
Certificate fingerprints:
MD5:  AB:FA:18:2B:BC:FF:1B:67:E7:69:07:2B:DB:E4:C6:D9
SHA1: 2E:98:D4:4B:E0:E7:B6:73:55:4E:5A:BE:C1:9F:EA:9B:71:18:60:BB
SHA256: F3:54:FB:67:80:10:BA:9C:3F:AB:48:0B:27:83:58:BB:3D:22:C5:27:7D:
F4:D1:85:C4:4E:87:57:72:2B:6F:27
Signature algorithm name: SHA1withRSA
Version: 3
Trust this certificate? [no]:  (yes)
Certificate was added to keystore
C:\Program Files\Java\jre7\lib\security>
```

Importing the Root Certificate to the Browser

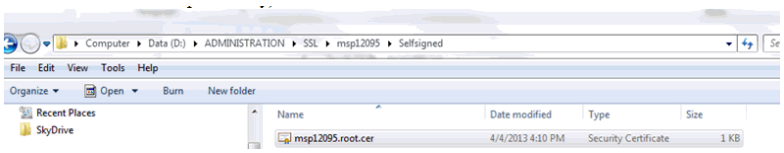
You need to add the signed Weblogic server certificate in the browser to avoid certificate verification error, if the Root Certificate is not in that list of trusted CAs.

Importing the Root Certificate through Internet Explorer

Perform the following steps to import the Root Certificate through Internet Explorer:

1. Copy the Root Certificate file to the workstation.
2. Rename the file to fa_root_cert.cer (this is a quick and easy way to associate the file with the Windows certificate import utility).

Figure 4–2 Importing the Root Certificate File to the Workstation



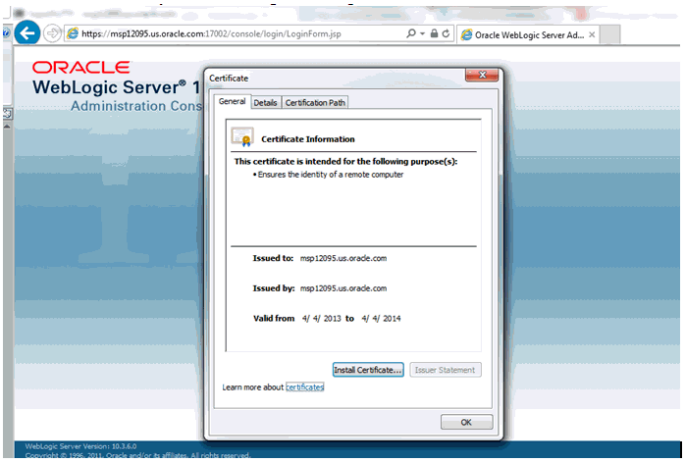
3. Select the file.
4. Click **Install Certificate** and click **Next**.
5. Select **Place all certificates in the following store** and click **Browse**.
6. Select **trusted Root Certification Authorities** and click **OK**.
7. Click **Next**.
8. Click **Finish** and then **Yes** at the Security Warning prompt.
9. Click **OK** to close the remaining open dialog boxes.

Importing the Root Certificate through Mozilla Firefox

Perform the following steps to import the Root Certificate through Mozilla Firefox:

1. Start Mozilla Firefox.
2. Select **Tools** > **Options** from the main menu.
3. Click **Advanced** >**Encryption** tab >**View Certificates**.
4. In Certificate Manager, click the **Authorities** tab and then the **Import** button.
5. In the Downloading Certificate dialog, choose **Trust this CA to identify websites** and click **OK**.
6. Click **OK** in Certificate Manager.
7. Open a browser and test the URL using the SSL port.

Figure 4–3 Importing the Root Certificate File through Mozilla Firefox



Secure Cookies

Cookies should be set to secure by default. To verify this has been correctly set, do the following:

1. Enable SSL in the environment.
2. Check the weblogic.xml. the cookie-secure element should be set to **true.**

```
<session-descriptor>
    <cookie-secure>true</cookie-secure>
</session-descriptor>
```

3. If the element is not set to true and the value has to be changed:

a. Redeploy the <app>.ear file.

b. Restart the services.

Changes to Web Application Descriptor

Secure transport configuration is default for SIM. To verify the default is correctly set:

1. Open the deployment descriptor of the Service Workspace, which has the jersey servlet configured.
2. Verify the transport-guarantee element is set to to CONFIDENTIAL.
3. The Application will be deployed to secure the ReST Services as a one way SSL as follows:

```
<security-constraint>
    <web-resource-collection>
      <web-resource-name>Workflow Actions</web-resource-name>
   <url-pattern>/services/private/*</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
    </web-resource-collection>
 <auth-constraint>
       ..
 </auth-constraint>
 <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
 </user-data-constraint>
  </security-constraint>
```

Note: An SSL connection needs to be used to ensure information being sent is not compromised, especially authentication credentials. If SSL is not used, the user credentials get passed with BASE-64 encoding which does not encrypt the credentials and would be a hole in security.

Launching Issues with RPM

For launching errors of RPM in the Java console, see the following example:

```
Caused by: java.net.ConnectException: t3s://apphost2:17012: Destination
unreachable; nested exception is:
javax.net.ssl.SSLKeyException: [Security:090542]Certificate chain received from
apphost2 - 10.141.13.195 was not trusted causing SSL handshake failure. Check the
certificate chain to determine if it should be trusted or not. If it should be
trusted, then update the client trusted CA configuration to trust the CA
certificate that signed the peer certificate chain. If you are connecting to a WLS
server that is using demo certificates (the default WLS server behavior), and you
want this client to trust demo certificates, then specify
-Dweblogic.security.TrustKeyStore=DemoTrust on the command line for this client.;
No available router to destination
at weblogic.rjvm.RJVMFinder.findOrCreateInternal(RJVMFinder.java:216)
at weblogic.rjvm.RJVMFinder.findOrCreate(RJVMFinder.java:170)
at weblogic.rjvm.ServerURL.findOrCreateRJVM(ServerURL.java:153)
at
weblogic.jndi.WLInitialContextFactoryDelegate.getInitialContext(WLInitialContextFa
ctoryDelegate.java:352)
... 27 more
```

The reason could be SSL Handshake failing between Desktop client and the RPM server. Try importing the root certificates in local client JRE (see the steps as provided in Importing the Root Certificate in Local Client JRE section). In case this fails, try disabling hostname verification to NONE for SSL Configuration of the managed server where RPM is deployed. See Disabling Hostname Verification section. This will require restart of the RPM managed server.

Disabling Hostname Verification

The hostname verification ensures that the hostname in the URL to which the client connects matches the hostname in the digital certificate that the server sends back as part of the SSL connection. However, in case SSL handshake is failing due to inability to verify hostname this workaround can be used.

> **Note:** Disabling hostname verification is not recommended on production environments. This is only recommended for testing purposes. Hostname verification helps to prevent man-in-the-middle attacks.

Perform the following steps to disable the hostname verification for testing purposes:

1. Go to **Environment** > **Domain** > **Servers** > **AdminServer**.
2. Click the **SSL** tab.
3. Click **Advanced**.
4. On Hostname Verification, select **NONE**.
5. Save and activate changes.
6. On the Node Manager startup script, look for JAVA. Add the following line:

   Dweblogic.nodemanager.sslHostNameVerificationEnabled=false

   After this change, the script should look as follows:

```
JAVA_OPTIONS="-Dweblogic.nodemanager.sslHostNameVerificationEnabled=false
${JAVA_OPTIONS}"
cd "${NODEMGR_HOME}"
set -x
if [ "$LISTEN_PORT" != "" ]
 then
   if [ "$LISTEN_ADDRESS" != "" ]
```

7. Restart Node manager.

Verifying the Certificate Content

In situations where the certificate expires or belongs to a different hosts, the certificates become unusable. You can use the keytool utility to determine the details of the certificate. The certificates should be renewed or new certificates should be obtained from the appropriate certificate authorities, if the certificates expire.

Example:

```
apphost1:[10.3.6_apps] /u00/webadmin/ssl> keytool -printcert -file cert.cer
Certificate[1]:
Owner: CN=apphost1, OU=<department>, O=<company>,L=<city>,ST=<state or province>,
C=<country>"
Issuer: CN=Oracle SSL CA, OU=Class 3 MPKI Secure Server CA, OU=VeriSign Trust
```

```
Network, O=Oracle Corporation, C=US
Serial number: 0078dab9f1a5b56e2cd6g92a3987296
Valid from: Thu Oct 11 20:00:00 EDT 2012 until: Sat Oct 12 19:59:59 EDT 2013
Certificate fingerprints:
         MD5:  2B:71:89:11:01:40:43:FC:6F:D7:FB:24:EB:11:A5:1C
         SHA1:
DA:EF:EC:1F:85:A9:DA:0E:E1:1B:50:A6:8B:A8:8A:BA:62:69:35:C1
         SHA256:
C6:6F:6B:A7:C5:2C:9C:3C:40:E3:40:9A:67:18:B9:DC:8A:97:52:DB:FD:AB:4B:E5:B2:56:47:E
C:A7:16:DF:B6
         Signature algorithm name: SHA1withRSA
         Version: 3

Extensions:
```

Verifying the Keystore Content

Keystores are repository of the certificates. If you face issues related to SSL Certificates, you need to check the certificates which are available in the keystore. You need to import the certificates if they are not missing. The keytool command provides the list of the certificates available.

Example:

```
$ keytool -v -list -keystore /u00/webadmin/product/jdk/jre/lib/security/cacerts
$ keytool -v -list -keystore /u00/webadmin/product/10.3.X_APPS/WLS/wlserver_
10.3/server/lib/apphost1.keystore
```

Integration Issues

Oracle Retail applications can be deployed across different hosts and behind network firewalls. Ensure firewalls are configured to allow TCPS connections to enable secure communications among integrated application.

Secured applications using signed certificates need to use same secured protocols for communication. Ensure that all the communicating applications use the same protocol.

For more information on steps to specify secured protocol, see Enforcing Stronger Encryption in WebLogic section of Chapter 1.

Communicating applications using signed certificates may need to verify the incoming connections. Root certificates should be available in the keystores of the applications to verify the requests from different host. It is important to import all the root certificates in the keystores of all communicating applications. For information on steps to import the root certificate in local client JRE, see Importing the Root Certificate in Local Client JRE section.

Errors in WLS_FORMS

When you try to restart the WLS_FORMS managed server in Oracle Forms installation after configuring for secure setup (enabling SSL), the managed server startup logs shows the error as shown in Example 4–1. To resolve, ensure that Additional configuration for WLS_FORMS (For forms server) in Pre-installation - Steps for Secured Setup of Oracle Retail Infrastructure in WebLogic of Chapter 1have been completed. The startup shows the errors in the logs as shown in the example, when you try to restart the WLS_FORMS managed server in Oracle Forms installation after configuring for security.

Example 4–1 WLS_Forms startup error

```
Feb 6, 2013 6:05:40 AM EST> <Notice> <Cluster> <BEA-000133> <Waiting to
synchronize with other running members of cluster_forms.>
<Feb 6, 2013 6:06:10 AM EST> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to ADMIN>
<Feb 6, 2013 6:06:10 AM EST> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to RESUMING>
<Feb 6, 2013 6:06:10 AM EST> <Error> <Cluster> <BEA-003111> <No channel exists for
replication calls for cluster cluster_forms>
<Feb 6, 2013 6:06:10 AM EST> <Critical> <WebLogicServer> <BEA-000386> <Server
subsystem failed. Reason: java.lang.AssertionError: No replication server channel
for WLS_FORMS
java.lang.AssertionError: No replication server channel for WLS_FORMS
        at
weblogic.cluster.replication.ReplicationManagerServerRef.initialize(ReplicationMan
agerServerRef.java:128)
        at
weblogic.cluster.replication.ReplicationManagerServerRef.<clinit>(ReplicationManag
erServerRef.java:84)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:186)
        at
weblogic.rmi.internal.BasicRuntimeDescriptor.getServerReferenceClass(BasicRuntimeD
escriptor.java:469)
        Truncated. see log file for complete stacktrace
>
<Feb 6, 2013 6:06:10 AM EST> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to FAILED>
<Feb 6, 2013 6:06:10 AM EST> <Error> <WebLogicServer> <BEA-000383> <A critical
service failed. The server will shut itself down>
<Feb 6, 2013 6:06:10 AM EST> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to FORCE_SHUTTING_DOWN>
<Feb 6, 2013 6:06:11 AM> <FINEST> <NodeManager> <Waiting for the process to die:
28209>
<Feb 6, 2013 6:06:11 AM> <INFO> <NodeManager> <Server failed during startup so
will not be restarted>
<Feb 6, 2013 6:06:11 AM> <FINEST> <NodeManager> <runMonitor returned, setting
finished=true and notifying waiters>
```

Ensure you have completed the steps mentioned in Additional Configuration for WLS_FORMS (For forms server) section of Chapter 1.

HTTPS Service Encountering Redirect Loop After Applying Policy A

The proxy server access enters into a redirect loop, if the services are secured with policy A (username token over SSL), and the deployment is in a cluster. The access to such services does not work.

Perform the following workaround through SB Console, for services that are secured with HTTPS:

1. Click **Resource Browser**.
2. Click **Proxy Services under Resource Browser**.
3. Click **Create under Change Center** to start a session.
4. For each of the SSL secured proxy services, perform the following steps:
    1. Click the proxy service you want to change.
    2. Click **Edit** next to **HTTP Transport Configuration**.

3. Uncheck **HTTPS Required** check box.
4. Click **Last**.
5. Click **Save**.

5. Click **Activate** and then **Submit**.

5

Importing Topology Certificate

Implementation of SSL into the Oracle Retail deployment is driven by mapping the SSL certificates and wallets to various participating components in the topology.

Importing Certificates into Middleware and Repository of Oracle Retail Applications

Table 5–1 describes the trust stores to be updated while confirming the certificates imported into middleware and repository of Oracle Retail applications. Ensure you have updated the given trust stores with the signed (either self signed or issued by certifying authority) certificates

---

Note: In Table 5–1, the *root.cer are the public key certificates and the *server.cer are the private key certificates.

Table 5–1 Importing Topology Certificate

		Java app-host		Forms app-host		RIB app-host		BIPublisher-host		OID-host	Client-host	
Component	**Certificates**	**Java app -Managed server**	**Java app-JAVA cacerts**	**Forms app - Managed server**	**Forms app-JAVA cacerts**	**RIB app-Managed server**	**RIB app-JAVA cacerts**	**BIPublisher - Managed server**	**BIPublisher - JAVA cacerts**	**Wallet**	**Browser**	**Client-JAVA cacerts**
Java.app	appserver.cer	Yes	No	No	No	No	No	No	No	No	No	No
Java.app	approot.cer	Yes	Yes	No	No	No	Yes	No	Yes	Yes	Yes	Yes
Forms.app	fmserver.cer.	No	No	Yes	No	No	No	No	No	No	No	No
Forms.app	fmroot.cer	No	No	No	Yes	No	No	No	Yes	Yes	Yes	Yes
RIB.app	ribserver.cer	No	No	No	No	Yes	No	No	No	No	No	No
RIB.app	ribroot.cer	No	Yes	No	No	Yes	Yes	No	No	No	Yes	Yes
BI Publisher	biserver.cer	No	No	No	No	No	No	Yes	No	No	No	No
BI Publisher	biroot.cer	No	Yes	No	Yes	No	No	Yes	Yes	No	Yes	Yes
OID	oidcer.cer	No	No	No	No	No	No	No	No	Yes	No	No
OID	oidroot.cer	No	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes

6

Using Self Signed Certificates

Self signed certificates can be used for development environment for securing applications. The generic steps to be followed for creating self signed certificates and configuring for use for Oracle Retail application deployment are covered in the subsequent sections.

The following topics are covered in this chapter:

Creating a Keystore through the Keytool in Fusion Middleware (FMW) 11g

Perform the following steps to create a keystore through the keytool in Fusion Middleware (FMW) 12c:

1. Create a directory for storing the keystores.

   $ mkdir ssl

2. Run the following to set the environment:

   $ cd $MIDDLEWARE_HOME/user_projects/domains/<domain>/bin

   $ . ./setDomainEnv.sh

   Example:

   ```
   apphost2:[10.3.6_apps] /u00/webadmin/product/10.3.6/WLS/user_
   projects/domains/APPDomain/bin> . ./setDomainEnv.sh
   apphost2:[10.3.6_apps] /u00/webadmin/product/10.3.6/WLS/user_
   projects/domains/APPDomain>
   ```

3. Create a keystore and private key, by executing the following command:

   ```
   keytool -genkey -alias <alias> -keyalg RSA -keysize 2048 -dname <dn> -keypass
   <password> -keystore <keystore> -storepass <password> -validity 365
   ```

   Example:

   ```
   apphost2:[10.3.6_apps] /u00/webadmin/ssl> keytool -genkey -alias apphost2
   ```

```
-keyalg RSA -keysize 2048 -dname "CN=<Server Name>,OU=<Organization Unit>,
O=<Organization>,L=<City>,ST=<State>,C=<Country>" -keypass <kpass> -keystore
/u00/webadmin/ssl/apphost2.keystore -storepass <spass> -validity 365

apphost2:[10.3.6_apps] /u00/webadmin/ssl> ls -ltra
total 12
drwxr-xr-x 18 webadmin dba 4096 Apr  4 05:31 ..
-rw-r--r--  1 webadmin dba 2261 Apr  4 05:46 apphost2.keystore
drwxr-xr-x  2 webadmin dba 4096 Apr  4 05:46 .
apphost2:[10.3.6_apps] /u00/webadmin/ssl>
```

Exporting the Certificate from the Identity Keystore into a File

Perform the following steps to export the certificate from the identity keystore into a file (for example, pubkey.cer) run the following command:

$ keytool -export -alias selfsignedcert -file pubkey.cer -keystore identity.jks -storepass <password>

Example:

```
apphost2:[10.3.6_apps] /u00/webadmin/ssl> keytool -export -alias apphost2 -file
/u00/webadmin/ssl/pubkey.cer -keystore /u00/webadmin/ssl/apphost2.keystore
-storepass <spass>
Certificate stored in file </u00/webadmin/ssl/ropubkey.cerot.cer>
apphost2:[10.3.6_apps] /u00/webadmin/ssl> ls -l
total 8
-rw-r--r-- 1 webadmin dba 2261 Apr  4 05:46 apphost2.keystore
-rw-r--r-- 1 webadmin dba  906 Apr  4 06:40 pubkey.cer
apphost2:[10.3.6_apps] /u00/webadmin/ssl>
```

Importing the Certificate Exported into trust.keystore

Perform the following steps to import the certificate you exported into trust.keystore run the following command:

$ keytool -import -alias selfsignedcert -trustcacerts -file pubkey.cer -keystore trust.keystore -storepass <password>

Example:

```
apphost2:[10.3.6_apps] /u00/webadmin/ssl> keytool -import -alias apphost2
-trustcacerts -file pubkey.cer -keystore trust.keystore -storepass <spass>
Owner: CN=apphost2, OU=<Organization Unit>, O=<company>,L=<city>,ST=<state or
province>, C=<country>
Issuer: CN=apphost2, OU=<Organization Unit>, O=<company>,L=<city>,ST=<state or
province>, C=<country>
Serial number: 515d4bfb
Valid from: Thu Apr 04 05:46:35 EDT 2013 until: Fri Apr 04 05:46:35 EDT 2014
Certificate fingerprints:
         MD5:  AB:FA:18:2B:BC:FF:1B:67:E7:69:07:2B:DB:E4:C6:D9
         SHA1: 2E:98:D4:4B:E0:E7:B6:73:55:4E:5A:BE:C1:9F:EA:9B:71:18:60:BB
         Signature algorithm name: SHA1withRSA
         Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
apphost2:[10.3.6_apps] /u00/webadmin/ssl>
```

Configuring WebLogic

You need to enable SSL for WebLogic server's Admin and managed servers by following the steps as provided in Configuring the Application Server for SSL section of Chapter 1.

Configuring Nodemanager

You need to secure the Node manager by following the steps in Securing Nodemanager with SSL Certificates section.

Importing Self Signed Root Certificate into Java Virtual Machine (JVM) Trust Store

In order for the Java Virtual Machine (JVM) to trust in your newly created certificate, import your custom certificates into your JVM trust store.

Perform the following steps to import the root certificate into JVM Trust Store:

1. Ensure that JAVA_HOME has been already set up.
2. Run the following command:

   $keytool -import -trustcacerts -file rootCer.cer -alias selfsignedcert -keystore cacerts

   Example:

```
apphost2:[10.3.6_apps] /u00/webadmin/product/jdk1.1.7_
30.64bit/jre/lib/security> keytool -import -trustcacerts -file
/u00/webadmin/ssl/root.cer -alias apphost2 -keystore
/u00/webadmin/product/jdk1.6.0_30.64bit/jre/lib/security/cacerts -storepass
[spass default is changeit]
Owner: CN=apphost2, OU=<Organization Unit>, O=<company>,L=<city>,ST=<state or
province>, C=<country>"
Issuer: CN=apphost2, OU=<Organization Unit>, O=<company>,L=<city>,ST=<state or
province>, C=<country>"
Serial number: 515d4bfb
Valid from: Thu Apr 04 05:46:35 EDT 2013 until: Fri Apr 04 05:46:35 EDT 2014
Certificate fingerprints:
         MD5:  AB:FA:18:2B:BC:FF:1B:67:E7:69:07:2B:DB:E4:C6:D9
         SHA1: 2E:98:D4:4B:E0:E7:B6:73:55:4E:5A:BE:C1:9F:EA:9B:71:18:60:BB
         Signature algorithm name: SHA1withRSA
         Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
apphost2:[10.3.6_apps] /u00/webadmin/product/jdk1.6.0_
30.64bit/jre/lib/security>
```

Disabling Hostname Verification

This section has been covered under Disabling Hostname Verification section.

Converting PKCS7 Certificate to x.509 Certificate

Certificate authorities provide signed certificates of different formats. However, not all formats of certificates can be imported to Java based keystores. Hence the certificates

need to be converted to usable form. Java based Keystores supports x.509 format of certificate.

The following example demonstrates converting certificate PKCS 7 to x.509 format:

1. Copy the PKCS 7 certificate file to a Windows desktop.
2. Rename the file and provide .p7b extension.
3. Open the .p7b file.
4. Click the plus (+) symbol.
5. Click the Certificates directory.

   An Intermediary certificate if provided by CA for trust.

   Note: If an Extended Validation certificate is being converted you should see three files. The End Entity certificate and the two EV intermediate CA's.

6. Right click on your certificate file.
7. Select All Tasks > Export.
8. Click **Next**.
9. Select Base-64 encoded X.509 (.cer) > click Next.
10. Browse to a location to store the file.
11. Enter a File name.

    For example, MyCert. The .cer extension is added automatically.
12. Click **Save**.
13. Click **Next**.
14. Click **Save**.

The certificate can be now imported into java based keystores.

Example:

```
apphost1:[10.3.6_apps] /u00/webadmin/ssl> keytool -import -trustcacerts -alias
apphost1 -file /u00/webadmin/ssl/cert-x509.cer -keystore
/u00/webadmin/product/jdk/jre/lib/security/cacerts
Enter keystore password: [default is changeit]
Certificate was added to keystore
apphost1:[10.3.6_apps] /u00/webadmin/ssl>
```

Part II

Oracle Retail Store Inventory Management

The following chapters provide guidance for administrators, developers, and system integrators who securely administer, customize, and integrate the Oracle Retail Warehouse Management System (SIM) application.

Part II contains the following chapters:

- Overview of Store Inventory Management Security
- Security Features
- Administration

Additional information may also be found in the I*nstallation Guide for the Oracle Retail Store Inventory Management*.

7

Overview of Store Inventory Management Security

This chapter discusses the security architecture and provides guidance for securing the Oracle Retail Store Inventory Management (SIM) application. Although each retailer must determine the detailed security methods that best suit its organization, this chapter outlines best practice for securing Store Inventory Management.

General Security Considerations

When considering how to make SIM secure, general security principles should be bourn in mind.

Software and Patches

Oracle Software

Oracle releases Critical Patch Updates for security related issues on a regular schedule four times a year. For more urgent vulnerabilities, Oracle issues Security Alerts for fixes deemed too critical to wait for the next Critical Patch Update. Information on these patches is available on the Oracle website and is also distributed by e-mail. It is strongly recommended that these patches be applied as soon as possible in order to address the identified vulnerabilities.

Third Party Software

Oracle software is often used in conjunction with third party software such as operating systems or drivers. It is strongly recommended that users regularly check for patches or updates that address security vulnerabilities in this software.

Users should also regularly stay up to date with security information and alerts which can report on vulnerabilities in third party operating systems, algorithms or recommended configurations.

Reducing the Scope for Security Breaches

Security fundamentals should be applied during security planning and while managing activities across all security related components.

General Principles

When hardening security for a SIM deployment, the complete application architecture and operational environment should be considered. See the relevant security documentation for detailed information on securing system components such as

operating systems, Java runtime environment, WebLogic application server, Oracle database server, Oracle Internet Directory.

The Java runtime environment should be maintained using the current Oracle security baseline. The client machine should have the correct certificates installed and Java runtime configuration for executing a signed trusted WebStart application launched using a JNLP file over SSL communication.

Securing the Environment

The environment can be made secure by minimizing the attack surface to reduce attack vectors. Methods of achieving this include:

- The client machine, application server machine, database server machine, and LDAP server machine should all be protected by firewalls and use separate user accounts and credentials.
- The environment can be further secured by removing or disabling all unnecessary components. For example, if the wireless server is not used then it should be shut down and removed post installation.
- As WebLogic server supports one active security realm per domain it is important to ensure that all applications deployed to the same domain share the same identity management solution. For example, if multiple applications use different identity stores then they should be deployed to separate domains with their respective security providers.

Separating Components

The severity of potential security breaches can be limited by separating the different system components. Hosting the application and database servers on two different machines would mean a security breach on one machine would not necessarily lead to a security breach on the other. For example, if the batch client WebLogic server was compromised, by using a separate user with access limited to SIM services, other web logic resources such as the database server would remain protected.

The separation of duties with respect to resources provides the opportunity to implement layered security, often referred to as defense-in-depth. As SIM uses a multi-tier architecture it is recommended to secure each layer separately. Although this increases the security complexity it improves the applications resilience against different forms of attack and reduces the risk of a single point of failure.

For example, the batch client should use an operating system user that does not have Administrator's privileges. That user should only have the access rights required to execute the batch client runtime, read and write batch files, network access limited to the SIM server. The WebLogic user account should not have Administrator's privileges and should only have access to the SIM server deployment.

Network Access

Access to the network should be restricted as much as possible. For example, use of network address white lists, firewalls, software/hardware VPN, encrypted connections, and user access restrictions can be used to limit access to the deployment at the network level.

User Access

SIM supports several user security modes and single sign on (SSO) authentication by using the WebLogic security services.

Identity management is the management of users and their associated authentication, authorisation and privileges within a software deployment. Correct configuration and management of those identity management solutions is important to maintaining a secure application deployment

User accounts should be carefully configured to provide the least amount of privilege required to perform their specific operations. This includes users and access to all areas of the application infrastructure. User privileges should be maintained actively and reviewed periodically to determine if changes are required.

System security should be continually monitored and maintained during operation. There are many potential causes for security breaches, even in secure systems, such as undiscovered vulnerabilities, technology advancements, user account abuse or theft. In order to quickly and effectively detect intrusions and mitigate risk it is important to audit activity and regularly monitor audit records.

Handheld Devices

The SIM PC and wireless handheld clients are intended for use in a retail store environment. Due to the risk of unauthorized users gaining physical access to the client device additional security measures are recommended. This includes device access restrictions such as a screen lock based on inactivity protected by password or two factor authentication. For example, the PC client machine should be configured to lock the screen after a period of inactivity and require user password authentication to be unlocked.

Wireless handheld client software is provided by Wavelink. This third party software renders SIM screens and connects to the Wavelink wireless server.

Secure Shell (SSH) is a protocol developed for transmitting private information over the Internet. SSH encrypts data that is transferred over the Telnet session. The Telnet Client supports multiple SSH versions and will automatically select the most secure protocol that the SSH server supports.

In SIM 15.0 the Wavelink client/server supports TLS transport security. The SIM wireless server should be configured for TLS security in production deployments.

Installation

This section covers the installation of SIM.

Pre-Installation

Before installation a deployment plan should be developed that considers system security in addition to application operation. There are many components and features to examine when hardening system security. By using formal planning with flowcharts and checklists there is less risk of mistakes and overlooking security vulnerabilities.

When installing each infrastructure component, such as the operating system or application server, the appropriate security documentation should be reviewed. Ensure that these components are securely configured and use appropriate security features, such as password policies and encryption.

Security options such as SSL or TLS should be required and set up using appropriate signed certificates and trusted certificate authority.

Installation

> **Note:** For detailed instructions on installing the SIM application, see the SIM Implementation Guide.

When installing the SIM application the available security features for all installed components should be enabled. For example, SSL should be enabled for all resources, and web service security policies should be applied.

It is important to only install necessary components and to disable any unused features during configuration.For example, if RIB is not used then it should be disabled during installation.

Separate user accounts should be used for each component configured by the installer. The user accounts should have access restricted to the requirements of the functions they will carry out. They should use credentials that satisfy robust password policies.

Depending on the selected identity management solution, the appropriate security providers must be installed for SIM to perform authentication and authorization operations.

- If the external (OID or LDAP) or a hybrid user security mode is used then the WebLogic OID or LDAP authentication provider must be configured for the WebLogic domain.
- If the internal (database) or a hybrid user security mode is used then the SIM database authentication provider must be installed and configured for the WebLogic domain.
- If SSO authentication is used then the SIM SSO authentication provider must be installed and configured for the WebLogic domain.

It is recommended to install only the required security providers and remove or disable any unused providers.

For a clustered installation it is strongly recommended to use an external credential store (database or LDAP) for Oracle Fusion Middleware security services.

Post Installation

After installing the product the deployment and environment security should be reviewed.

All unnecessary components should be removed or disabled, including but not limited to resources, services, application features, weak protocols and insecure access points. Also ensure that appropriate file restrictions have been applied to protect any sensitive information and limit access to minimum requirements.

If an internal user security mode is used then application roles and internal user accounts should be set up. The installer creates an internal installation user who has access to configuring application settings, role management, and user account management. Once users have been set up, the internal installation user should be deleted from the system

Role Based Security and User Management

It is good practice to determine application roles based on organization requirements and use a fine grained modular structure for role composition. As users can be granted

multiple roles using these fine grained roles, this can help limit access to required functions while reducing the frequency of role management activities.

Additional declarative security can be provided using deployment descriptors, so only specific users are allowed to invoke certain EJBs.

For detailed information about SIM application role based security and user management see the *Oracle Retail SIM Implementation Guides*.

Web Based Security

Web services provided and consumed by SIM can be configured with security policies by the installer. These web services are designed to participate in Retail Service Backbone (RSB) flows which support two distinct Oracle WebLogic WS-Policy configurations. These are referred to as Policy A and Policy B.

On the provider side of the communication, Policy A and Policy B are configured using one or more Oracle WebLogic WS-Policy configurations defined in the xml files included in Oracle WebLogic:

- Policy A
  - Description:

    Message must be sent over SSL and requires authentication of a plain text UsernameToken.
  - Configuration:

    Wssp1.2-2007-Https-UsernameToken-Plain.xml
- Policy B
  - Description:

    Message body must be encrypted and signed, and requires authentication of an encrypted UsernameToken.
  - Configuration:
    * Wssp1.2-2007-Wss1.1-UsernameTokenPlain-EncryptedKey-Basic128.xml
    * Wssp1.2-2007-EncryptBody.xml
    * Non-RSB Web ServicesWssp1.2-2007-SignBody.xml

For information on where these policies are set for web service consumers see *Appendix: SIM Application WebLogic Server Installer Screens* in the *Oracle Retail SIM Implementation Guides.*

For detailed information on configuring the Oracle WebLogic security policies, see the *Oracle Retail Service Backbone Security Guide*.

8

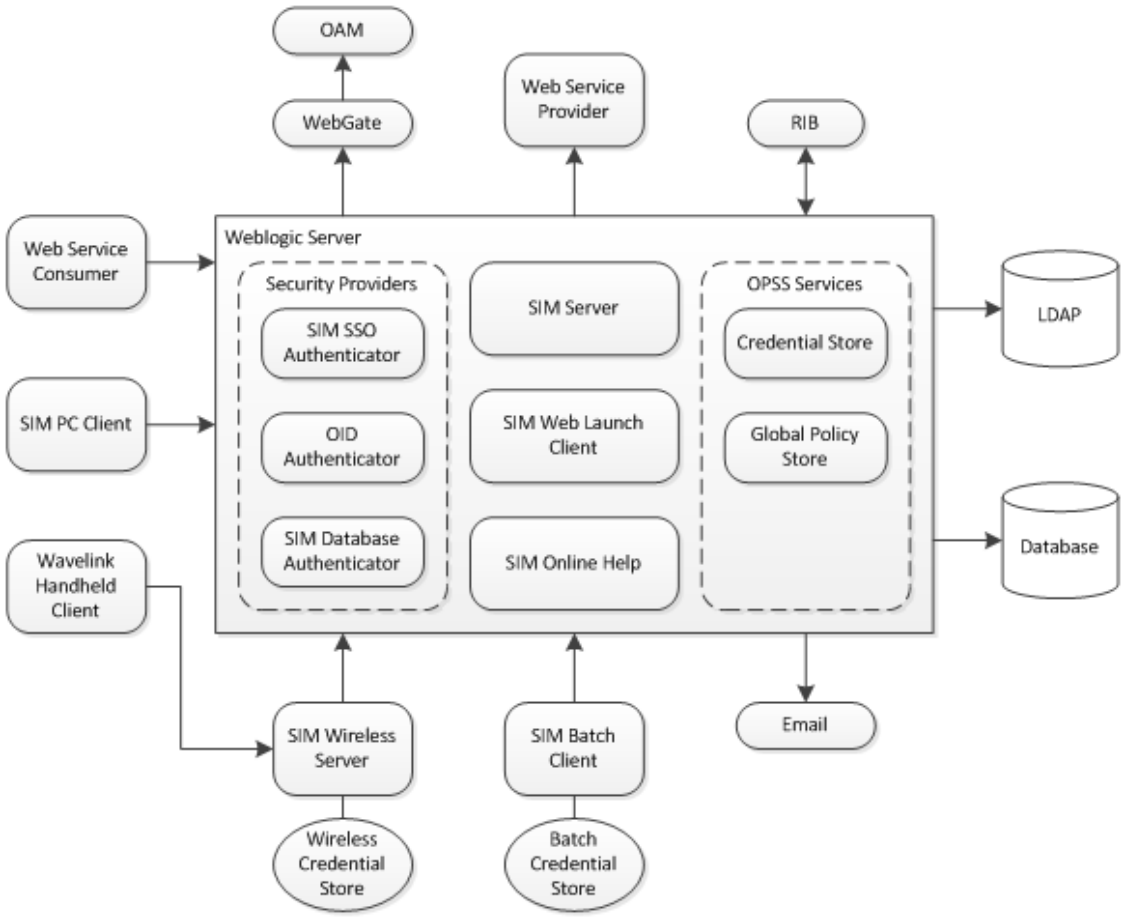
Security Features

This section describes the security features built into SIM.

Overview of Security Features

The SIM application uses a multi-tier architecture. It is important to understand how it is structured when considering application infrastructure security.

Figure 8–1 SIM Architecture Logical Model



The SIM application is deployed to an instance of a WebLogic application server running on a private network. SIM requires the WebLogic domain to be created with

the Java Required Files (JRF) extension, which includes the Oracle Platform Security Services (OPSS). OPSS is a security framework that provides security to Oracle Fusion Middleware and associated applications.

SIM uses the OPSS credential store framework and Java authorization functionality to manage sensitive information related to application security. For example, SIM stores encryption keys and integration user credentials in the credential store.

The OPSS configuration for the WebLogic domain can be managed through Oracle Enterprise Manager. For detailed information on OPSS, see the Oracle Fusion Middleware Application Security Guide.

SIM supports multiple user security modes that affect user authentication and authorization. Based on the selected identity management solution and organization requirements, the appropriate user security mode should be used.

- The default and recommended user security mode is external authentication and authorization, which allows LDAP to store and manage user data.
- The internal authentication and authorization mode allows the SIM application to manage the user data, which is stored in the database.

There is a hybrid user security mode that uses a combination of LDAP and the database for storage and management of user data. This mode allows for an externally managed identity store while retaining application functionality for user management of internal data. When accessing both external and internal data, the external system is prioritized before the internal system.

If a hybrid mode is used then it is recommended to disable the cache or reduce it to the minimum acceptable value.Authentication and authorization capabilities are configurable through security provider setup and user permissions to produce alternative restricted variations.

For further information on the user security modes and configuration, see the Oracle Retail SIM Implementation Guide Volume 1.

Dependent Applications

Information on securing the dependent applications can be found in the following security guides:

- Oracle Database 12c 2 Security Guide.
- Oracle Application Security Guide.
- WebLogic 12.1.3 Security Guide

Technical Overview of Security Features

The application security features are organized into authentication, authorization, audit, and user management sections.

Authentication

In order for a remote client to connect to the SIM application, user authentication is required. SIM supports authentication for SSO, LDAP, and database security. This is handled by WebLogic security authentication providers.

WebLogic security authentication providers are based on the JAAS framework, which allows the providers to be ordered and apply control flags that determine how the

overall sequence behaves. If a user cannot be authenticated then the client will not be allowed to connect to the SIM application.

When the user is successfully authenticated by one of the authentication providers, the user is then processed for authorization to grant access to resources and application features.

Authorization

User authorization is handled by both the WebLogic application server and the SIM application. After successful authentication the security providers will have specified privileges for the session such as a user and associated roles/groups. The deployed resources can be configured within WebLogic application server to restrict access based on those privileges. This allows for fine grained access control of resources. It is recommended that application users be given access only to the SIM functionality required for their tasks.

The SIM application handles business authorization for an authenticated user during the login process. Application features are restricted using a role based access control system. Roles are managed in the SIM database to define sets of permissions that are required to perform business operations associated with organization roles. User access is also restricted by stores and allows roles to be assigned per store.

Role and store assignments for users are managed in the selected identity store (LDAP or database). When a user logs into a store the authorization process validates role and store assignments and applies the resulting collection of permissions to the user session. Access to application features is limited to the set of permissions granted to the user session.

In addition, access to application and server resources is restricted to users that belong to specific groups, such as the administrator, security ops, MPS ops groups.

For detailed information on SIM role based security and permissions, see the Oracle Retail SIM Implementation and Operation Guides.

Audit

SIM provides basic audit capabilities for business processes. Audit records are held in the SIM database as either generic activity history or specific history records for certain business areas such as item UIN history.

User Management

The SIM client application can be used to manage user data, role, store and group assignments that are managed internally (database). External user data (LDAP, SSO) is not managed by SIM and must be performed using appropriate organization resources.

For detailed information on user management of internal SIM users, see the Oracle Retail SIM Implementation and User Guides.

Encryption and Hashing

Security components within SIM are designed to allow for extension and customization. This allows cryptography operations to be implemented as needed. The SIM standard security implementation uses the Java Cryptography Extension (JCE) framework to protect sensitive and security related information.

The algorithms parameters used for cryptography are configurable through configuration files. For detailed information on algorithm options and parameters, see the Oracle JCE documentation.

SIM uses hashing for storing user passwords in the database when an internal authentication security mode is used. The hashing can only be performed in one direction, which ensures that passwords cannot be reverse engineered.

The hashing is performed using an encryption key that is stored separately in the credential store. Passwords can only be validated using the same encryption key. If user passwords are migrated between databases then the encryption key must also be migrated between credential stores. If an encryption key is lost or changed then all existing passwords in the database will fail authentication.

By default the application will automatically generate a random encryption key during initialization if none exists. Manual key management can be performed by updating the credential store.The user password cryptography configuration is located in the *server.cfg* file, located in *sim-server-resources.jar*.

For detailed information on this configuration file, see the *Oracle Retail Store Inventory Management Operations Guide*.

SIM uses encryption and hashing for generation and validation of SSO tokens when SSO authentication is used. The token is generated using an encryption key that is stored separately in the credential store.

By default the application will automatically generate a new random encryption key during initialization, which simplifies key management. For manual key management the automatic generation can be disabled through configuration, allowing the encryption key to be manually managed by updating the credential store. The SSO token cryptography configuration is located in the *web-launch.properties* file, located in *sim-client.war*.

For detailed information on this configuration file, see the *Oracle Retail Store Inventory Management Operations Guide*.

9
Administration

This section covers the administration of SIM security.

Roles and Permissions

SIM uses role based security to control user access to application features and functionality.

Permissions represent authorized access to specific operations or functionality. Roles are created to represent job functions that correspond to specific levels of authority, and are assigned one or more permissions.

Users are assigned one or more roles in order to grant access to functionality related to their job. A user's authorized permission set is determined by the union of the permission sets for each authorized role. Role assignments can be constrained by start and end dates.

Users are also assigned one of more stores in order to grant access to specific stores. These store assignments are required for a user to log into the store, or even access role assignments for the store.

Super users are an exception to this rule. They have access to all stores, but still require role assignments to gain access to functionality.

For detail information on role based security and user management, see the *Oracle Retail SIM Implementation Guide*.

Common Application Administration

SIM uses the OPSS credential store framework for managing sensitive information related to application security. Although some data such as encryption keys may be automatically managed by the application, some of the credentials are configured by the installer, such as RIB integration credentials. The credential store data can be managed through either OPSS scripts or using Oracle Enterprise Manager.

For detailed information on the OPSS credential store framework, see the *Oracle Fusion Middleware Application Security Guide*.

Log files are generated by WebLogic application server and SIM application.

- The WebLogic log files are configured and managed by the application server and contain infrastructure information.
- The SIM application log files are produced according to the SIM log configuration and contain application operation information. The default configuration

generates log files in a log directory in the WebLogic domain directory structure, but shares the same OS user and file access conditions as the WebLogic log files.

It is recommended to restrict access to log files purely to administrators and the WebLogic application server process owner.

Generally SIM does not limit the number of concurrent sessions for users. The SIM client application includes a feature that prevents more than one instance of the client to be launched concurrently on a single machine, although it does not prevent the same user from logging in on different machines or devices simultaneously. This option is enabled by default but can be disabled through configuration in the *client.cfg* file, located in *sim-client-resources.jar*.

For detailed information on this configuration file, see the *Oracle Retail Store Inventory Management Operations Guide*.

Session timeouts can be controlled by configuring resources in the application deployment and WebLogic server. The application uses WebLogic default timeouts with some exceptions for long running EJB services, such as batch operations. These timeout overrides can be found in the *weblogic-ejb-jar.xml* deployment descriptor file, located in *sim-server.ear*.

It is recommended to use the minimum appropriate timeout values to reduce the impact of denial of service attacks to resource availability. However, these timeout values should not be set so short as to interfere with the operations of legitimate users.

Extending/Customization

SIM security features have been designed to allow for extension and customization. This includes configuration options such as algorithms and parameters. It also allows custom implementations of security components to be used.

Custom implementations are developed using the same customization patterns found elsewhere in SIM. This involves extending or replacing factory implementations that provide custom implementations of factory built objects. For customization of classes that are not built by a factory, the implementation class name is configured in the respective configuration file.

For detailed information on configuration options and application customization, see the *Oracle Retail Store Inventory Management Operations* and *Implementation Guide*s.

References

The following documents give more information:

- *Oracle Retail Store Inventory Management 15.0 Release Note*s.
- *Oracle Retail Store Inventory Management 15.0 Installation Guide*.
- *Oracle Retail Store Inventory Management 15.0 User Guide*.
- *Oracle Retail Store Inventory Management 15.0 Implementation Guides*.
- *Oracle Retail Store Inventory Management 15.0 Operations Guide*.