Oracle® Retail Advanced Science Engine

Security Guide

Release 14.0

E50592-01

December 2013

ORACLE®

Oracle® Retail Advanced Science Engine Security Guide, Release 14.0

E50592-01

Primary Author: Judith Meskill

Contributing Author: Vlad Yelizarov

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

3 Securing the WebLogic Server

4 Securing the Database

5 Using Self-Signed Certificates

6 Troubleshooting

A Appendix: Secure Services and Protocols

Send Us Your Comments

Oracle® Retail Advanced Science Engine Security Guide, Release 14.0

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

This document serves as a guide for administrators, developers, and system integrators who securely administer, customize, and integrate the application. For more information on installing the application, refer to the *Oracle Retail Modeling Engine Installation Guide* and the *Oracle Retail Assortment and Space Optimization 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 application 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.

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 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 application documentation set:

- *Oracle® Retail Advanced Science Engine Implementation Guide*
- *Oracle® Retail Assortment and Space Optimization Installation Guide*

- *Oracle® Retail Assortment and Space Optimization User Guide*
- *Oracle® Retail Modeling Engine Installation Guide*
- *Oracle® Retail Modeling Engine User Guide*

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, 14.0) or a later patch release (for example, 14.0.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

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/technology/documentation/oracle_retail.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.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

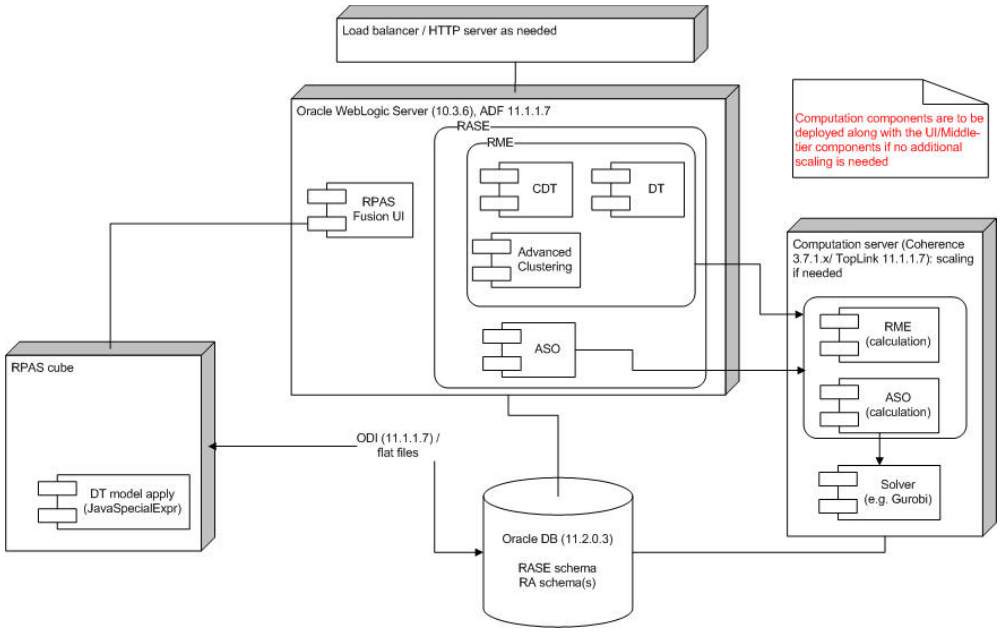
Overview

This chapter provides an overview of the security features in the application. It includes the following sections:

Physical Deployment Model

The following figure illustrates the physical deployment model of the application:

Figure 1–1 Physical Deployment Model



The Web-based application user interface is accessed using the Microsoft Internet Explorer on Windows-based systems. The retailer is responsible for applying the necessary security patches to the Web browser and operating system (Microsoft Windows).

The typical configuration of the application runs on multiple servers: one for the Oracle WebLogic Server 11g Release 1 (10.3.6) extended to use Oracle Application Development Runtime 11.1.1.7 that hosts the application and one for the Oracle database.

The application, the ETL scripts, and the export scripts are hosted on the WebLogic server, and the database instances are installed on the server running the Oracle Database 11g Release 2 (11.2.0.3). The retailer is responsible for applying any critical patch updates released for the server hardware, application server, and database.

The database server contains database schemas and includes a set of PL/SQL procedures. All the application stages are executed on the database server. ETL scripts are used for the loading, staging, and transformation of the application data input. Each of the application stages that are executed on the Database Server performs operations on a set of input tables and generates results to output tables. The output from one stage is provided to the next stage in succession as each stage is executed. The application output files are generated for import by the subscribing Oracle Retail applications, including Category Management and RDF, from the Database Server. It is the retailer's responsibility to ensure the integrity of the import data and output files and to secure access to the import files on the servers.

The application produces files used by subscribing Oracle Retail applications. If these files are modified, the resulting calculations may be inaccurate. For this reason, the files should be put in directories where appropriate permissions have been applied and unauthorized access is prevented.

Oracle Retail Advanced Science Engine has separate modules: Retail Modeling Engine is supported on the cloud but Assortment and Space Optimization is not. Ensure that the server systems running the database and application server and the client systems are located within a secured network. For scalability reasons, it is possible to add (optional) computation node(s), connected into a Oracle Coherence cluster. Oracle Coherence communication can be secured by performing the relevant configuration.

Dependent Applications

Security Guides for dependent applications can be found at the following links:

- Oracle Database 11g Release 2 (11.2.0.3):

  http://docs.oracle.com/cd/E11882_01/server.112/e10575.pdf

- Oracle WebLogic Server 11g Release 1 (10.3.6):

  http://docs.oracle.com/cd/E21764_01/web.1111/e14529/security.htm

Security Features Overview

Caution: Oracle is not responsible for the security compliance of any product customization performed by a retailer, system integrator, or reseller.

The relevant security features fall into one or more of the following categories. For information on these categories, see the following sections:

- Securing the Application
- Vulnerability Management

Securing the Application

Securing access to the application against malicious attacks and auditing secure events are accomplished with passwords, additional testing of Web applications, and additional examination of source code.

Default Accounts and Passwords

The application does not contain any default accounts, user IDs, or passwords. The application uses the database schema user names created when setting up the application database. For more information, refer to the relevant installation guides.

Passwords

Password policy settings for database schema access are configured through the database. As such, passwords must conform to the database password policies set for your business needs. For more information on setting up and maintaining password policies, see the Guidelines for Securing Passwords section in the *Oracle Database 11g Release 2 Security Guide*.

The application utilizes WebLogic and ADF Security API's for authentication and authorization. The application contains the necessary application roles and WebLogic-integrated LDAP allows mapping of application roles to enterprise roles. Fusion Middleware Control (aka Enterprise Manager) can be used to perform the mapping

Development and Testing

Since the application is a Web-based application, it is subjected to additional testing and scrutiny. The application is tested with tools designed to subject the applications to known attack methods in an effort to identify areas of vulnerability. In addition, the source code for the application undergoes static code analysis.

Oracle Retail application maintains a team of individuals who are responsible for monitoring newly discovered security vulnerabilities to see if they affect the applications. Additionally, each release undergoes intrusion and penetration testing.

Product updates and patches are produced and tested by the Development team and are made available to retailers through secure download through the My Oracle Support Web site.

Oracle Retail application development standards mandate that all code is peer reviewed before it is promoted to the code base. Retailers and system integrators who are customizing or extending any of the applications, should consider implementing their own security coding standards and review processes. If customization is done, care should be taken when adding sensitive information to the system or data files.

The application does not prevent the use of network address translation, port address translation, traffic filtering devices, anti-virus protection, or encryption. Also, it does not interfere with the installation of patches or updates. Due to the nature of subtle incompatibilities between application server implementations, retailers are advised to test the latest application server updates with the application prior to putting those updates into production environments.

The application implements a uniform time out of the application when left unattended. If the application is left unattended for longer defined time, the application will time out. The user returning to the application will be taken back to the login screen and will have to log in again. The transaction that the user was

executing is cancelled and the user will have to re-execute the transaction. The time-out value can be configured in the web.xml file for the application.

Tools

The application uses the Fortify 360 tool to scan for security issues. As with any tool, the output of this tool should be analyzed in detail since the output may contain false positive warnings.

You can use any tools that you choose. No recommendation of the following tool is intended or implied.

Fortify 360 is a tool that analyzes software for vulnerabilities. The static analysis component examines an application's source code for potentially exploitable vulnerabilities. The dynamic analysis component identifies vulnerabilities that can be found only when an application is running. All vulnerabilities can be ranked according to their relevance.

Fortify can be found at the following Web site:

http://www.fortify.com/

Vulnerability Management

The Open Web Application Security Project (OWASP) periodically lists its top 10 vulnerabilities for web applications. You should pay special attention to these vulnerabilities when coding or testing modifications to the application. The OWASP Top Ten can be found at the following Web site:

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Injection Flaws

The application is tested with automated tools to help detect code vulnerabilities. The following are two possible SQL injection vulnerabilities:

- A PreparedStatement object is created based on a non-constant string of SQL. This may or may not include user-modifiable parameters from the Web layer. If no parameters are included in the SQL, then no SQL injection vulnerability exists for that query.
- A non-constant string of SQL is passed directly to an execute method of a Statement object. This may or may not include any user-modifiable parameters.

The safe and preferred way to execute SQL is to create a PreparedStatement object, apply the parameters, and call the execute method on that object. A PreparedStatement object has the SQL statement inside it compiled before any parameters are applied. Doing so means that a malicious parameter cannot change the SQL query that will be run.

Injection flaws are not limited to SQL injection. The application validates and escapes dynamic data to prevent HTML, XML, and other types of injection.

Insecure Communications

From OWASP (http://www.owasp.org/index.php/Top_10_2007-A9):

Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications. Encryption (usually SSL) must be used for all authenticated connections, especially Internet-accessible Web pages, and backend connections as well. Otherwise, the application will expose an authentication or session token. In addition, encryption should be used whenever sensitive data, such as credit card or

health information, is transmitted. Applications that fall back or can be forced out of an encrypting mode can be abused by attackers.

All communication between a browser and application is transmitted over SSL by default.

Improper Error Handling

The application provides error messages to the user that convey the nature of the error without compromising important system information. Systems integrators and retailers who extend or modify the product should ensure that any new error conditions created by the modifications do not provide more information than necessary.

Cross Site Scripting

The application prevents cross-site scripting attacks through the proper use of JSP tags and output escaping of dynamic data.

Improper Access Control

Insecure object references occur when an application exposes key data directly or indirectly to the user. This could be in the form of a primary key value in a hidden field or shown in a URL. A malicious user could modify that data in an attempt to access objects that the user would not normally be able to access. The application considers the current user's access prior to allowing the user access to objects.

Failure to control URL access could allow a user with insufficient permissions to perform an operation that the user would otherwise be unable to accomplish, if the user knew the URL to call and the data to pass to the application for that operation. The application prevents users from accessing URLs that they do not have permission to view.

Securing the Application Environment and Configuration

Securing the application environment and configuration covers the following area:

- Database

Database

Once sensitive data is stored in a database, that data must be protected from unauthorized access. Oracle Retail provides the following recommendations on how to protect that data:

- Access to the stored procedures used in the data purge scripts should be restricted.
- Authentication to the database should be done with a different user ID than authentication to the applications.

The application does not populate the database with any pre-defined users. An administrative user is created during installation.

2

Secure Configuration

This chapter serves as a guide for administrators and anyone installing the product to securely configure Oracle Retail Advanced Science Engine (ORASE).

The chapter begins with the operating system and moves through the supporting middleware to the application and its connections with other resources. It includes the following sections:

- Operating System Considerations
- Infrastructure and Middleware Considerations
- Application Configuration Considerations

> **Note:** The options set by default for the installer are the most secure selection. If you choose to not use any of the default selections, you need to consider the implications of that change on the security of your installed product.

Intended Audience

This chapter is intended for security administrators and anyone who will install, deploy, and configure ORASE. These users typically perform the following tasks:

- Install and deploy the applications
- Configure the applications
- Apply patches to the applications

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

Operating System Considerations

This section describes any specific considerations related to the supported operating systems.

ORASE does not rely on insecure services or protocols. If the retailer or systems integrator customizes or extends the applications, these extensions must not rely on insecure services or protocols.

This section includes information on:

- Oracle Linux-Based Systems
- Oracle Solaris-Based Systems

Oracle Linux-Based Systems

Ensure that you have installed all the critical patch updates (CPU) and patches available for Oracle Linux. Critical patch updates and patch sets for Oracle products are made available on the My Oracle Support Web site along with documentation or instructions on how you can install them.

For more information on securing Oracle Linux-based systems, refer to the Oracle Linux documentation.

Oracle Solaris-Based Systems

Ensure that you have installed all the critical patch updates (CPU) and patches available for Oracle Solaris. Critical patch updates and patch sets for Oracle products are made available on the My Oracle Support Web site along with documentation or instructions on how you can install them.

For more information on securing Oracle Solaris-based systems, refer to the *Oracle Solaris Security Guidelines*.

IBM AIX-Based Systems

Ensure that you have installed all the security patches and patch sets available for IBM-AIX. Security patches and patch sets for IBM AIX may be made available on the IBM Support Portal along with documentation or instructions on how you can install them.

For more information on securing IBM AIX-based systems, refer to the *IBM AIX Security Guide*.

Additional Resources

The Center for Internet Security has published benchmarks for securing your systems at the operating system level. You can find the benchmark for some of the operating systems at the following link:

`http://benchmarks.cisecurity.org/en-us/?route=downloads.browse.category.benchmarks.os`

Infrastructure and Middleware Considerations

This section describes any specific considerations related to the supported infrastructure and middleware products (such as databases and application servers). This section includes the following sections:

Database

Do not store sensitive data on Internet-accessible systems. For example, your application server and database server must not be on the same physical server. ORASE does not require the database server and application server to be hosted on the same physical server machine.

For information about secure configuration of Oracle Database, see the *Oracle Database 2 Day + Security Guide*. The guide is available at the following link on the Oracle Technology Network Web site:

http://docs.oracle.com/cd/E11882_01/server.112/e10575/toc.htm

Application Server

For information on the secure configuration of Oracle WebLogic Server, see the following documentation available at the following links on the Oracle Technology Network Web site:

- Security for Oracle WebLogic Server 11g Release 1:

  http://docs.oracle.com/cd/E21764_01/web.1111/e14529/security.htm

- *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*:

  http://docs.oracle.com/cd/E21764_01/web.1111/e13705/toc.htm

Application Configuration Considerations

This section describes any specific application-related configuration recommended for ORASE. It includes the following sections:

- Integration with Other Applications
- Scripts and Command Line Utilities

Integration with Other Applications

ORASE integrates with Oracle Retail RDF, Oracle Retail Category Management, Oracle Retail Retail Analytics, and Oracle Retail MSM applications through the use of File Copy and Shared Database schemas.

Note: File Copy refers to sharing data between applications by copying files to a file system.

Scripts and Command Line Utilities

ORASE includes scripts and utilities that can be used after the installation to stage, load, and transform data. Ensure that access to such scripts and utilities are set up based on the business need.

3

Securing the WebLogic Server

The application supports the use of Oracle WebLogic Server 11g Release 1 (10.3.6), extended to use Oracle Application Development Runtime Release 11.1.1.7. The WebLogic server must be secured using the security recommendations provided in the *Oracle Fusion Middleware Information Roadmap for Oracle WebLogic Server 11g Release 1*. This chapter provides additional specific guidance for securing the WebLogic server for use with the application. It includes the following sections:

Install Patch Set Updates

Before you start setting up the database, ensure that you have installed all the critical patch updates (CPU) and patches for the WebLogic server. Critical patch updates and patch sets for Oracle products are made available on the My Oracle Support Web site along with documentation or instructions on how you can install them.

Setting Up a Secure WebLogic Domain

When setting up the WebLogic domain, set up the following configuration parameters to ensure a secured configuration:

Administrative User Account

In a secured configuration, the WebLogic server administrative user names and passwords must not use any default or predictable values, such as weblogic1, welcome1, weblogic, and so on. When setting up the WebLogic domain, ensure that you use non-standard user names and passwords.

Operating System User Account

When the WebLogic server and domains are installed and set up, ensure that they are not running under the *root* operating system user account. You can check for this by reviewing the permissions on the WebLogic server files and folders. None of these objects should be owned by the *root* operating system user.

Listen Port Configuration

Once the WebLogic domain for the application is created, ensure that you manually disable the HTTP port and enable the HTTPS port. This ensures that only a secure channel is used for accessing the application. This should be done for two reasons:

- User credentials over HTTP are sent in clear-text and anyone listening in on the network will see the user names and passwords.
- Disabling the HTTP protocol ensures the application is only accessed via HTTPS for all URLs.

You must also ensure that the secure HTTPS port number is changed to a non-default value. This value must be environment-specific, non-standard, and not easily predictable.

For more information on configuring the listen ports, refer to the *Oracle Fusion Middleware Administrator's Guide*.

Setting Up Keystores

A Java keystore (JKS) is a secured database that stores keys and certificates for an organization. It is used to achieve authentication, integrity, and privacy within in a network. The WebLogic server uses JKS keystores for applications deployed in the WebLogic server.

By default, the WebLogic server is configured with a demo identity keystore and trust keystore. These keystores must not be used in a production environment. You must create your own keystores in the production environment and set up the WebLogic server to use them.

This section describes how you can create your own keystore and trust store. It also describes the necessary configuration steps to set up the WebLogic server with your JKS keystore. It includes the following sections:

- Setting Up Keystores and Trust Stores
- Associating the Keystore and Trust Store with WebLogic Server

Setting Up Keystores and Trust Stores

To set up your own keystore and trust store: **<change commands>**

1. Create a new directory called keystores in your application deployment and navigate to this directory.
2. Run the following command to create your keystore and certificate:

```
keytool -genkey -keyalg RSA -alias RASEselfsigned -keystore RASEkeystore.jks
-storepass  password -validity 360 -keysize 2048
```

3. Run the following command to export your certificate from the keystore:

```
keytool -export -alias RASEselfsigned -keystore RASEkeystore.jks -rfc -file
RASEselfsigned.cer
```

4. Run the following command to create a trust store and add your certificate to the list of trusted certificates:

```
keytool -import -alias RASEselfsigned -file RASEselfsigned.cer -keystore
RASEtrust.ts -storepass password
```

Once you are done, there will be three files in the keystores folder—the keystore, the trust store, and the certificate.

> **Note:** In the code snippets above, the alias name (**RASEselfsigned**), keystore name (**RASEkeystore**), and trust store (**RASEtrust**) are used for illustration purposes in this document. You may choose to set up names you want.

Associating the Keystore and Trust Store with WebLogic Server

To associate the keystore and trust store with the WebLogic server instance:

1. Log on to the **WebLogic Server Administration Console**.
2. From the **Domain Configurations** section, click **Servers**, under the **Environment** category. The **Summary of Servers** page appears.
3. On the **Summary of Servers** page, under the **Configuration** tab, click the relevant server used for ORASE. The **Settings** page for the server appears.
4. On the **Settings** page, in the **Configuration** tab, click the **Keystores** tab.
5. In the **Keystores** tab, click the **Change** button next to the **Keystores** field.

> **Note:** You may need to lock the configuration for editing.

6. From the drop-down list, select **Custom Identity and Custom Trust**.
7. Click **Save**.
8. Enter relevant information in the following fields:
   - **Custom Identity Keystore** – Specify the location of the keystore file (.jks). For example, /u00/oracle/rase133/keystore/RASEkeystore.jks.
   - **Custom Identity Keystore Type** – Specify the type of the keystore. Enter the text jks.
   - **Custom Identity Keystore Passphrase** – Specify the password associated with the keystore (set up when you created the keystore).
   - **Confirm Custom Identity Keystore Passphrase** – Specify the same password again to confirm.
   - **Custom Trust Keystore** – Specify the location of the trust store file (.ts). For example, /u00/oracle/rase133/keystore/RASEtrust.ts.
   - **Custom Trust Keystore Type** – Specify the type of the trust store. Enter the text jks.
   - **Custom Trust Keystore Passphrase** – Specify the password associated with the trust store (set up when you created the trust store).
   - **Confirm Custom Trust Keystore Passphrase** – Specify the same password again to confirm.
9. Click **Save** to save the entries on the **Keystores** tab.
10. Click the **SSL** tab and enter the relevant values in the following fields:
    - **Private Key Alias** – Specify the name of the keystore self-signed certificate. For example, RASEselfsigned.

- **Private Key Passphrase** – Specify the private key password associated with the certificate (set up when you created the certificate).
- **Confirm Private Key Passphrase** – Specify the password again to confirm.

11. Click **Save** and activate your configuration changes.
12. Restart the WebLogic server for the changes to take effect.

To see the changes to the keystores, ensure that the SSL is already enabled.

Configuring WebLogic Scripts in Order to Secure the Administration Server

Perform the following steps to configure the WebLogic scripts in order to secure the Administration Server:

1. Update the WebLogic startup/shutdown scripts with secured port and protocol to start/stop services.
2. Back up and update the following files in <DOMAIN_HOME>/bin with the correct administration 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 WebLogic"

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

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

3. Change the URLs as follows:

   Certificate was added to keystore

   t3s://apphost1:7002

   https://apphost1:7002

Adding a Certificate to the JDK Keystore for the Installer

You will need the Retail Application installer to run Java. If the Administration Server is secured using signed certificate, the Java keystore used to launch the installer must have the certificate installed.

If the installer is being run using JDK deployed at location /u00/webadmin/product/jdk, complete the following:

```
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 that you use a stronger encryption protocol in your production environment.

The following sections describe how to enable the latest SSL and cipher suites.

SSL Protocol Version Configuration

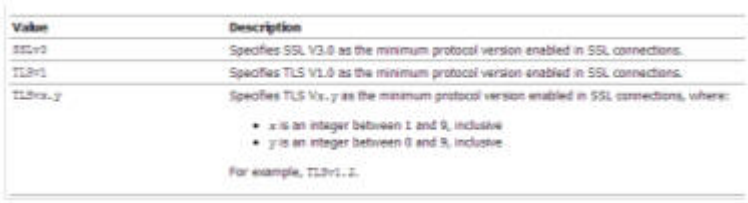
In a production environment, Oracle recommends Transport Layer Security (TLS) Version 1.1 or higher for sending and receiving messages in an SSL connection.

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 3–1 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 Vx.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:** If the protocol is set for Managed servers, it should be set for the 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 must 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 must ensure that the tag <ciphersuite> is added immediately after the tag <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

Complete the following steps to secure the Nodemanager with SSL certificates:

1. Navigate to <BEA_HOME>/wlserver_10.3/common/nodemanager and back up nodemanager.properties.
2. Add the following similar entries to nodemanager.properties:

   KeyStores=CustomIdentityAndCustomTrust

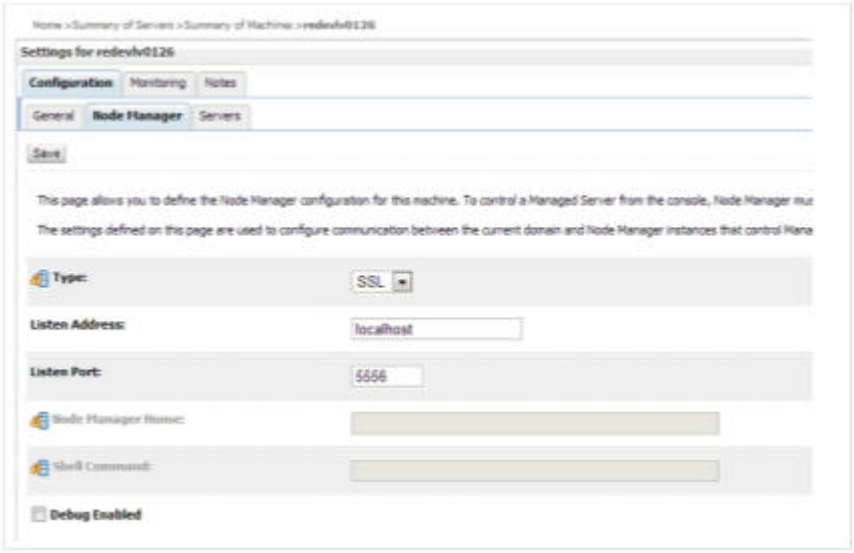
   CustomIdentityKeyStoreFileName=/u00/webadmin/ssl/hostname.keystore

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

   CusCustomIdentityPrivateKeyPassPhrase=[password to keystore, this will get encrypted]tomIdentityAlias=hostname

   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 3–2 Securing the Nodemanager



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

4

Securing the Database

The application supports the use of the Oracle Database 11g Release 2 (11.2.0.3). The database must be secured using the recommendations provided in the *Oracle Database 11g Release 2 Security Guide*. This chapter provides additional specific guidance for securing the database for use with the application. It includes the following sections:

Install Patch Set Updates

Before you start setting up the database, ensure that you have installed all the critical patch updates (CPU) and patches for the database. Critical patch updates and patch sets for Oracle products are made available on the My Oracle Support Web site along with documentation or instructions on how you can install them.

Application Schema Owners

The following recommendations should be considered for the schema owners:

- Database Administrators should create an individual schema owner for each database schema (specify application schemas).
- The schema owners should only have enough access privileges to install the application.

  For more information on creating database user accounts and the specific access privileges for the schema owners, see the section *Creating the Data User Accounts* in the chapter *Setting Up the Database* of the relevant installation guide.
- It is recommended that the user ID and password comply with the following policies:
  - Do not use group, shared, or generic accounts and passwords.
  - Require a minimum password length of at least seven characters.
  - Use passwords containing both numeric and alphabetic characters.
  - Do not allow an individual to submit a new password that is the same as any of the last four passwords used.

- Limit repeated access attempts by locking out the user ID after not more than six attempts.
- Set the lockout duration to 35 minutes or until an administrator enables the user ID.

Note: You can also choose to change user passwords at least every 90 days. In case you do choose to set this policy, ensure that the passwords set up in the connection pools for the application data sources in the WebLogic Server Administration Console are also updated to reflect the latest password. Once updated, the WebLogic server will need to be restarted for the changes to take effect.

Database Security Considerations

The following recommendations should be considered for the database:

- The database should be on its own dedicated server.
- The database server should be in a private network.
- The database server should be in a locked secure facility and inaccessible to non-administrator personnel.
- The database should only be accessed using trusted network hosts.
- The database server should have minimal use of ports and any communications should be under secure protocols.
- The database server should be behind a firewall.
- Any database user beyond the schema application owner should be audited.
- Only minimal rights should be granted to the owner of database processes and files such that only this owner has the right to read and write from the database related files and no one else has the capability to read and write from such files.

Special Security Options for Oracle Databases

Password policies can be enforced using database profiles. The options in the following table are based on version 11.2.0.3 of Oracle Database. The options can be changed using a SQL statement, for example:

```
alter profile appsample limit
```

Option	Setting	Description
PASSWORD_LOCK_TIME	30	Time account will be locked in minutes
FAILED_LOGIN_ATTEMPTS	4	Maximum number of login attempts before the account is locked
PASSWORD_GRACE_TIME	3	Number of days a user has to change an expired password before the account is locked
PASSWORD_REUSE_MAX	10	Number of unique passwords the user must supply before the first password can be reused

Option	Setting	Description
PASSWORD_VERIFY_FUNCTION	<routine_name>	Name of the procedure that can be created to ensure the password is acceptable

Password policies can be enforced using a password complexity verification script, for example:

```
UTLPWDMG.SQL
```

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 /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 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 name>)(PORT=2484))
        (ADDRESS=(PROTOCOL=tcp)(HOST=<dbserver name>)(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 /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 /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 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>)))
```

5. You need to set the properties as shown in Setting Properties, either as system properties or as JDBC connection properties.

Table 4–1 Setting 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 Oracle Wallet for Batch Script Execution

To configure Oracle Wallet for batch script execution, complete the following steps:

1. Create a wallet (for example, in /home/jdbc/wallet_db):

```
mkstore -wrl /home/jdbc/wallet_db -create
```

2. Store the credentials in the wallet:

```
mkstore -wrl /home/jdbc/wallet_db -createCredential RME_DB rme_schema
```

3. In the environment setup script, specify the wallet location:

```
export TNS_ADMIN=/home/jdbc/wallet_db
```

4. Create tnsnames.ora in the wallet location:

```
RME_DB=(DESCRIPTION=(ADDRESS_
LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_
NAME=SERVICE_0)))
```

5. Create sqlnet.ora in the wallet location:

```
WALLET_LOCATION=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=/home/jdbc/wallet_
db)))
SQLNET.WALLET_OVERRIDE=TRUE
SSL_CLIENT_AUTHENTICATION=FALSE
```

6. The persistence.xml file must be modified. Remove the user and password entries and change the url entry to match the wallet configuration:

```
<property name="javax.persistence.jdbc.url" value="jdbc:oracle:oci:/@RME_DB"/>
```

5

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 Retail application deployment are covered in the subsequent sections.

This chapter contains the following sections:

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

Complete the following steps to create a keystore through the keytool in Fusion Middleware (FMW) 11g:

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=apphost2,OU=RGBU, O=Oracle
   ```

```
Corporation,L=Minneapolis,ST=Minnesota,C=US" -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

Complete the following 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

Complete the following 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=RGBU, O=Oracle Corporation, L=Minneapolis, ST=Minnesota,
C=US
Issuer: CN=apphost2, OU=RGBU, O=Oracle Corporation, L=Minneapolis, ST=Minnesota,
C=US
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>
```

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.

Complete 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/jdk1.6.0_
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=<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 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>
```

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. So 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 CAs.

6. Right click on the 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>
```

6

Troubleshooting

This chapter covers common errors and issues and how to troubleshoot them. It contains the following sections:

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

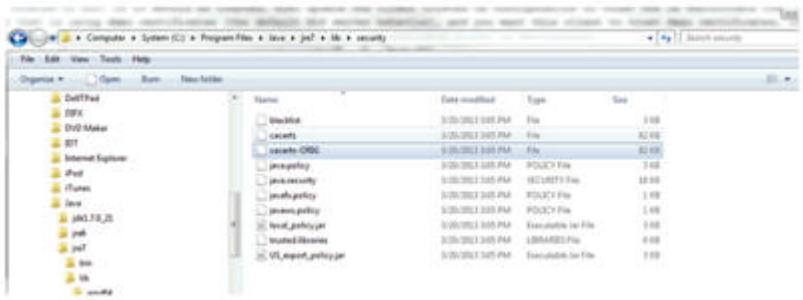
If customers are using certificates other than those provided by standard certificate authorities in a custom CA implementation, then the JRE used for launching the applications from local machines like laptops or desktops might display a different error messages.

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

Perform the following steps to import the root certificates:

1. Back up cacert at <JRE_HOME>/lib/security/cacert.

Figure 6–1 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 must add the signed WebLogic server certificate in the browser to avoid a certificate verification error if the Root Certificate is not in the list of trusted CAs.

Importing the Root Certificate through Internet Explorer

Complete 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 way to associate the file with the Windows certificate import utility).

Figure 6–2 Importing the Root Certificate File to the Workstation

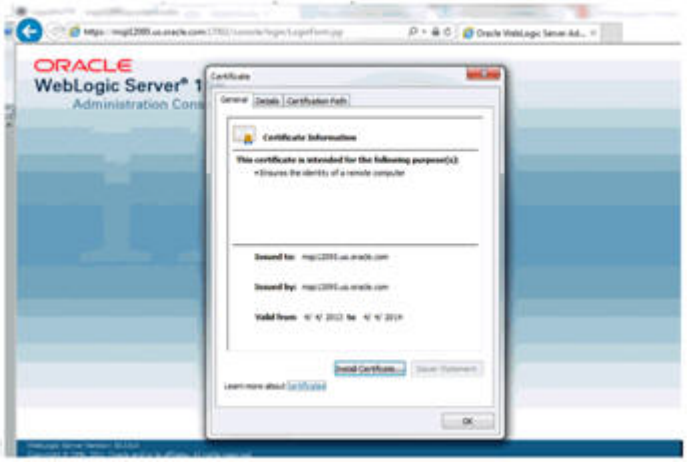
3. Select the file.
4. Click **Install Certificate** and click **Next**.
5. Select **Place all certificates in the following store** and click **Browse**.
6. Select **trusted Root Certification Authorities** and click **OK**.
7. Click **Next**.
8. Click **Finish** and then **Yes** at the Security Warning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

Importing the Root Certificate Through Mozilla Firefox

Complete 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 6–3 Import the Root Certificate File Through Mozilla



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. It is only recommended for testing purposes. Hostname verification helps to prevent man-in-the-middle attacks.

Complete 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 a certificate expires or belongs to a different host, the certificate become unusable. You can use the keytool utility to determine the details of the certificate. The certificate should be renewed or a new certificate should be obtained from the appropriate certificate authorities, if the certificate 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:EC:A7:16:DF:B6

Signature algorithm name: SHA1withRSA

Version: 3

Extensions:

Verifying Keystore Content

Keystores are repository of the certificates. When issues related to SSL Certificates exist, you should check the certificates that are available in the keystore. If the certificates are not missing, they should be imported. 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
```

HTTPS Service Encountering a Redirect Loop After Applying Policy A

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

Complete 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**.

A

Appendix: Secure Services and Protocols

In general, securing services and protocol requires retailers to take the following actions:

- Control physical and electronic access to the systems that handle sensitive data.
- Provide regularly scheduled auditing of network and network component activity.
- Deactivate unnecessary operating system components and securely configure those that remain active.

This appendix highlights some of the network and hardware security considerations for the application. It includes the following sections:

For information about Oracle Coherence, see http://docs.oracle.com/cd/E24290_01/coh.371/e22841.pdf

Securing the Network

Protecting the application data on the network is accomplished through the use of multiple security techniques. This is sometimes referred to as a Defense in Depth strategy, where each security technique helps to mitigate the risk of one component of the defense being compromised or circumvented. Depending upon the business and technological needs of each retailer, consider the following best practices for operating a network securely:

- Segment the network—The physical network is composed of isolated parts, divided along the different security and management needs of individual applications.

  The network configuration should include a private network for the application, making it impossible to connect to the Internet.

- Control access to routers and switches—Create a platform-specific minimum configuration standard for all routers and switches that follow industry best practices for security and performance.
- Utilize firewalls—Hardware firewalls should utilize explicit rules tuned to the services and ports needed by the applications served by the network.

- Secure the wireless network—Enforce encryption and require certificate-based authentication.
- Control physical access to networks and network devices.
- Use a centralized system for authentication and authorization that provides each user with unique and strongly protected credentials.
- Obscure the purpose of network resources through the use of naming conventions.
- Implement a strategy for monitoring and auditing network access and activity.

Resources

For more information on securing networks, see the following Web sites:

- http://www.microsoft.com/enterprise/industry/retail-hospitality/default.aspx#fbid=NzsOJCXDiJr
- http://www.novell.com/industries/retail/

Physical Security

Retailers must take precautions to ensure that any user with malicious intent cannot gain physical access to networks and devices. All equipment involved in the application activity must be physically secured, including cables and equipment housings. The client systems must be configured to automatically lock when left alone and must require a password that conforms to the password policy guidelines to unlock the register.

Audit and Monitoring

Systems running the database and application servers must routinely be audited for signs of compromise. Processes and procedures must exist to detect the installation and execution of unauthorized routines. Application and operating system logs should be fully utilized. Determining the cause of a compromise is extremely difficult without system activity details.

Equipment Storage and Disposal

Systems no longer in use, or temporarily stored, must be properly scrubbed of data. Your equipment vendor can provide the steps necessary to render the device data storage useless to an attack.