Oracle® Fusion Middleware

Administering Oracle Event Processing

12*c* Release (12.1.3)

E28536-05

October 2015

How to administer Oracle Event Processing applications and server clusters. Includes application deployment and Oracle Coherence, Jetty, JDBC and security configuration procedures.

ORACLE

Oracle Fusion Middleware Administering Oracle Event Processing, 12c Release (12.1.3)

E28536-05

Primary Authors: Madhubala Ponnekanti, Oracle® Corporation

Contents

10 Jetty

11 JMX

12 JDBC

13 HTTP Publish-Subscribe Server

14 Logging and Debugging

Part V Command Reference

A wlevs.Admin Command-Line Reference

Preface

This document describes how to configure and manageOracle Event Processing servers. Use Oracle Event Processing Visualizer for most administrative tasks. See *Using Visualizer for Oracle Event Processing*.

This administration guide describes command-line utilities and programmatic interfaces. The programmatic interfaces include management beans (MBeans) and interfaces to manage server group and domain membership. The interfaces enable developers to design and create a management console for their Oracle Event Processing installation to be used by their administrators. The command-line utilities are for administrators who prefer to conduct administrative tasks from the command line.

Audience

This section identifies the audience for whom the document is intended.

This document is intended for Oracle Event Processing server administrators.

Related Documents

For more information, see the following:

- Known Issues for Oracle SOA Products and Oracle AIA Foundation Pack at: http://www.oracle.com//technetwork/middleware/soasuite/documentation/soaknown-2644661.html.
- *Developing Applications for Oracle Event Processing*
- *Getting Started with Oracle Event Processing*
- *Getting Started with Oracle Edge Analytics*
- *Schema Reference for Oracle Event Processing*
- *Customizing Oracle Event Processing*
- *Using Visualizer for Oracle Event Processing*
- *Customizing Oracle Event Processing*
- *Developing Applications with Oracle CQL Data Cartridges*
- *Oracle CQL Language Reference for Oracle Event Processing*
- *Java API Reference for Oracle Event Processing*

- *Java API Reference for Oracle Edge Analytics*
- *Using Oracle Stream Explorer*
- *Getting Started with Oracle Stream Explorer*
- *Oracle Database SQL Language Reference* at: `http://docs.oracle.com/cd/E16655_01/server.121/e17209/toc.htm`
- SQL99 Specifications (ISO/IEC 9075-1:1999, ISO/IEC 9075-2:1999, ISO/IEC 9075-3:1999, and ISO/IEC 9075-4:1999)
- Oracle Event Processing Forum: `http://forums.oracle.com/forums/forum.jspa?forumID=820`

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.

What's New in This Guide

This guide has been updated for the 12*c* release. File names, screen shots, and text is updated to reflect new file names and file system structure. The following table lists other changes.

Sections	Changes Made
Entire Guide	**Product renamed to Oracle Event Processing.**
Network I/O Providers	Removed reference to several classes that are for internal use only.
Deployment Commands	Removed the `startLevel` command, which is not supported.

Part I

Overview

This part page contains the chapters that provide an overview.

Overview contains the following chapter:

- Introduction to Server Administration

1

Introduction to Server Administration

Oracle Event Processing server administration tasks involve creating domains and administering domains, servers, and applications. This guide describes the server administration tools you run from the command line to accomplish these tasks.

You can also use Oracle Event Processing Visualizer. Visualizer is a browser-based tool that enables you to administer Oracle Event Processing servers and domains, and to view, develop, configure and monitor aspects of Oracle Event Processing applications and security. See *Using Visualizer for Oracle Event Processing*.

This chapter includes the following sections:

See also *Java API Reference for Oracle Event Processing* for information about the Oracle Event Processing APIs described in this guide.

1.1 Server-Provided Services

An Oracle Event Processing server consists of logically related resources and services to which you deploy Oracle Event Processing applications. Services include:

- Network I/O: Server and client Internet Protocol (IP) port access, IPv4 and IPv6 support, and a variety of blocking and non-blocking network I/O providers.
- Security: Security services such as SSL, password stores, and authentication and authorization providers.
- Jetty: HTTP publish-subscribe server: To enable web clients to subscribe to channels and publish asynchronous messages to these channels over HTTP
- Java Management Extensions (JMX): Programmatic access to Oracle Event Processing server and application behavior.
- JDBC data sources: To access relational databases to store events for event record and playback, to access a table as an event source for Oracle CQL queries.
- HTTP publish-subscribe server: To push event messages to subscribed clients such as the Oracle Event Processing Visualizer and your own Web 2.0 applications.

- Logging: To monitor and troubleshoot server and application operation.

1.2 Server Domains

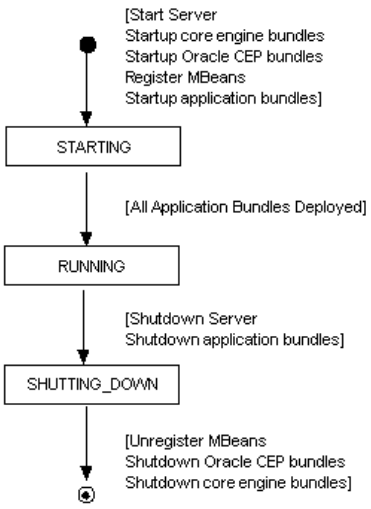
An Oracle Event Processing domain is the management unit of a set of one or more servers. There are two types of domain:

- Standalone-server domain: A domain that contains a single server. This is the type of domain that is created by default by the Configuration Wizard and is the starting point for a multiserver domain. See Standalone-Server Domains.
- Multiserver domain: A domain that contains two or more servers that share the same multicast address, multicast port, domain, and security provider. The multicast address, multicast port, and domain are configured in the `config.xml` file for each server in the domain. You can create server groups within a multiserver domain and deploy applications to each server in the specified server group. The servers within a multiserver domain can be located on the same computer or on separate computers.

1.3 Server Life Cycle

Figure 1-1 shows a state diagram for the Oracle Event Processing server life cycle. In this diagram, the state names (`STARTING`, `RUNNING`, and `SHUTTING_DOWN`) correspond to the `ServerRuntimeMBean.getState` method return values. These states are specific to Oracle Event Processing. They are not OSGi bundle states.

Figure 1-1 Server Life Cycle State Diagram



1.3.1 Server Startup Actions

After you start the Oracle Event Processing server, it performs the following actions:

1. Starts core engine bundles.
2. Starts Oracle Event Processing bundles and extension libraries.
3. Registers MBeans.
4. Oracle Event Processing server state is now `STARTING`.

5. Starts application libraries and then application bundles.
6. Oracle Event Processing server state is now `RUNNING`.

1.3.2 Server Shutdown Actions

After you shutdown the Oracle Event Processing server, it performs the following actions:

1. Oracle Event Processing server state is `SHUTTING_DOWN`.
2. Unregister `ServerRuntimeMBean`.

   Oracle Event Processing server ceases to have a state.
3. Shuts down Oracle Event Processing bundles.
4. Shuts down application bundles.
5. Shuts down core engine bundles.

1.4 Server Configuration

You can configure the server and configure applications deployed to the server and perform the asks statically or dynamically. Static configuration involves editing XML files. Dynamic configuration involves manipulating management beans (MBeans) with Oracle Event Processing Visualizer, the `wlevs.Admin` command-line tool, or programmatically with JMX APIs.

Static Configuration

There are some server configuration tasks that you can only perform statically, such as configuring Jetty.

To configure the server statically:

1. Stop the Oracle Event Processing server.
2. Edit the Oracle Event Processing server `config.xml` file located in the server's domain directory
3. Start the Oracle Event Processing server.

Dynamic Configuration

There are some server configuration tasks that you can perform dynamically using JMX and MBeans. In this case you do not have to manually stop and start the server for the changes to take effect.

After you deploy an application, you can dynamically change its configuration and the configuration of its individual components by manipulating the MBeans that the Oracle Event Processing server automatically creates for the application and its components. A typical task is to dynamically configure the Oracle CQL rules for the processors of a deployed application. You do this using Oracle Event Processing Visualizer, `wlevs.Admin` command-line utility, or JMX.

Related Information

For more information, see:

- Configure Servers

- *Using Visualizer for Oracle Event Processing*
- wlevs.Admin Command-Line Reference
- JMX .

1.4.1 Server Configuration Files

All server files are contained in a single server directory. The main server configuration file is `config.xml`. The `config.xml` file is where you configure the server services and specify the domain to which the server belongs.

By default, the Configuration Wizard creates server domains in the */Oracle/Middleware/my_oep/user_projects/domains* directory. The following list describes the important server files and directories that are in each domain:

- `deployments.xml`: An XML file that contains the list of applications, packaged as OSGi bundles, that are currently deployed to the Oracle Event Processing instance of this domain. You never update this file manually to deploy applications, but use the Deployer tool.
- `startwlevs.cmd`: A command file that you use to start an instance of an Oracle Event Processing server. The UNIX equivalent is `startwlevs.sh`.
- `stopwlevs.cmd`: A command file that you use to stop an instance of an Oracle Event Processing server. The UNIX equivalent is `stopwlevs.sh`.
- `config/config.xml`: An XML file that describes the configured services for the Oracle Event Processing server instance. Services include logging, debugging, Jetty Web Service, and JDBC data sources.
- `config/security*`: Files that configure security for the domain.
- `config/atnstore.txt`: A File that lists the configured users and user groups for the domain.

1.4.2 Edit the config.xml File

The most efficient and least error-prone way to configure an Oracle Event Processing server is to use one or more of the Oracle Event Processing administration tools described in Server Administration Tools.

Optionally, you can perform Oracle Event Processing server configuration by editing the Oracle Event Processing server `config.xml` file.

Caution:

If you update the `config.xml` file manually to change the configuration of an Oracle Event Processing server, you must restart the server for the change to take effect.

You can configure the following server objects and features using the `config.xml` file. The referenced sections describe the exact elements you must add or update:

- How the servers in a multiserver domain are configured together. This includes the multicast address and multicast port, the server groups, and so on. See:

  - Standalone-Server Domains
  - Multiserver Domains with Oracle Coherence
  - Multiserver Domains with Native Clustering
- Network I/O. See Network I/O.
- Security. See Security.
- Jetty, an open-source, standards-based, full-featured Java Web Server. See Jetty.
- JMX, required to use the Oracle Event Processing Visualizer, `wlevs.Admin` utility, and Deployer utility See JMX .
- JDBC data source, used to connect to a relational database. See JDBC.
- HTTP publish-subscribe server. See HTTP Publish-Subscribe Server.
- Logging and debugging properties of the server. By default, the log level is set to `NOTICE`. See Logging and Debugging.

The following example shows a sample `config.xml` that contains configurations for some of these services.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2007 sp2 (http://www.altova.com)-->
<n1:config
  xsi:schemaLocation="http://www.bea.com/ns/wlevs/config/server wlevs_server_config.xsd"
  xmlns:n1="http://www.bea.com/ns/wlevs/config/server"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <netio>
      <name>NetIO</name>
      <port>9002</port>
   </netio>
   <netio>
      <name>sslNetIo</name>
      <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
      <port>9003</port>
   </netio>
   <work-manager>
      <name>JettyWorkManager</name>
      <min-threads-constraint>5</min-threads-constraint>
      <max-threads-constraint>10</max-threads-constraint>
   </work-manager>
   <jetty>
      <name>JettyServer</name>
      <network-io-name>NetIO</network-io-name>
      <work-manager-name>JettyWorkManager</work-manager-name>
      <secure-network-io-name>sslNetIo</secure-network-io-name>
   </jetty>
   <rmi>
      <name>RMI</name>
      <http-service-name>JettyServer</http-service-name>
   </rmi>
   <jndi-context>
      <name>JNDI</name>
   </jndi-context>
   <exported-jndi-context>
      <name>exportedJndi</name>
      <rmi-service-name>RMI</rmi-service-name>
   </exported-jndi-context>
   <jmx>
      <rmi-service-name>RMI</rmi-service-name>
      <jndi-service-name>JNDI</jndi-service-name>
   </jmx>
   <ssl>
```

```
      <name>sslConfig</name>
      <key-store>./ssl/evsidentity.jks</key-store>
      <key-store-pass>
         <password>{Salted-3DES}j4XEtuXmmvEl4M/NInwq0A==</password>
      </key-store-pass>
      <key-store-alias>evsidentity</key-store-alias>
      <key-manager-algorithm>SunX509</key-manager-algorithm>
      <ssl-protocol>TLS</ssl-protocol>
      <enforce-fips>false</enforce-fips>
      <need-client-auth>false</need-client-auth>
   </ssl>
   <http-pubsub>
      <name>pubsub</name>
      <path>/pubsub</path>
      <pub-sub-bean>
         <server-config>
            <name>/pubsub</name>
            <supported-transport>
               <types>
                  <element>long-polling</element>
               </types>
            </supported-transport>
            <publish-without-connect-allowed>true</publish-without-connect-allowed>
         </server-config>
         <channels>
            <element>
               <channel-pattern>/evsmonitor</channel-pattern>
            </element>
            <element>
               <channel-pattern>/evsalert</channel-pattern>
            </element>
            <element>
               <channel-pattern>/evsdomainchange</channel-pattern>
            </element>
         </channels>
      </pub-sub-bean>
   </http-pubsub>
   <cluster>
      <server-name>productionServer</server-name>
   </cluster>
   <domain>
      <name>oep_domain</name>
   </domain>
```

1.4.3 Manage Configuration History

When you deploy an application to a server, the server creates a configuration history for the application. Any configuration changes you make to the application are recorded in this history. You can view and roll back (undo) these changes with Oracle Event Processing Visualizer or the `wlevs.Admin` tool.

1.4.4 Configure the Server bootclasspath

Use the `-Xbootclasspath` command set the search path for bootstrap classes and resources. For example, you can use this command to satisfy server dependencies beyond those set by the server configuration file (`config.xml`). You can also use this command to satisfy application and application library dependencies beyond those set by application import statements and found in the library and library extensions directories.

Configure the bootclasspath:

1. Change directories to the server directory of the domain for which you want to configure the bootclasspath.

The location is:

/Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<server_name>

2. In the server directory open the start script (`startwlevs.sh` or `startwlevs.cmd`, depending on your operating system in an editor.

3. Locate the following line:

```
"$JAVA_HOME/bin/java" $JVM_ARGS $JVM_D64 $DEBUG_ARGS -
Dwlevs.home="$USER_INSTALL_DIR" -jar "${USER_INSTALL_DIR}/bin/wlevs.jar" $ARGS
/wlevs.jar
```

4. Set the `-Xbootclasspath/a` option to the full path name of the native library you are going to use.

   For example, if you want to use the native library `mynativelib` located in Oracle Event Processing server directory `%USER_INSTALL_DIR%\bin`, update the `java` command in the start script as follows.

   The example is broken for readability. Put the full command on one line.

```
%JAVA_HOME%\bin\java -Dwlevs.home=%USER_INSTALL_DIR% -Dbea.home=%BEA_HOME%
-Xbootclasspath/a:\Oracle\Middlware\my_oep\oep\bin\mynativelib.jar
-jar "%USER_INSTALL_DIR%\bin\wlevs.jar"
-disablesecurity %1 %2 %3 %4 %5 %6
```

5. If Oracle Event Processing is running, restart it so it reads the new `java` option and data source information. See "Start and Stop Servers".

1.5 Server Administration Tools

This section describes the server administration tools that you can use to administer Oracle Event Processing servers, domains, and applications.

- Configuration Wizard. A Java application that you can invoke graphically to create and update Oracle Event Processing servers and domains. For the 12.1.3 release, the Configuration Wizard can generate an Oracle database configuration only. See Standalone-Server Domains.
- `wlevs.Admin` command-line utility. A Java application that you can invoke locally or remotely to perform a wide variety of Oracle Event Processing server, domain, and application administration tasks. See wlevs.Admin Command-Line Reference.
- Deployer command-line utility. A Java application that you can invoke locally or remotely to perform application deployment and application administration tasks. See Deployer Command-Line Reference.
- Security administration utilities. See Security Utilities Command-Line Reference.
- JMX. A set of standards-based interfaces that enable you to perform server, domain, and application administration tasks using JMX and management beans (MBeans). See JMX .

1.6 Server Administration Tasks

This section briefly describes some of the important server administration tasks.

Create Servers and Domains

The primary administrative task in setting up an Oracle Event Processing platform is creating and configuring the server domains. Oracle Event Processing supports standalone-server domains and multiserver domains.

For more information, see:

- Standalone-Server Domains
- Multiserver Domains with Oracle Coherence
- Multiserver Domains with Native Clustering

Update Servers and Domains

Once you create an Oracle Event Processing server and domain, you can update it to change its configuration or server group membership. See:

- Update a Standalone-Server Domain
- Update a Multiserver Domain
- Update a Multiserver Domain

Configure Servers

Once you create an Oracle Event Processing server and domain, you must configure the various services they provide. See:

- Network I/O
- Security
- Jetty
- Configure JMX
- JDBC
- HTTP Publish-Subscribe Server
- Logging and Debugging

Start and Stop Servers

After you have created an Oracle Event Processing domain along with at least a single server, you start a server instance so you can then deploy applications and begin running them. During upgrades and after some configuration changes, you must stop and start the Oracle Event Processing server. See:

- Start and Stop a Server in a Standalone-Server Domain
- Oracle Coherence: Start and Stop a Server in a Multiserver Domain
- Oracle Event Processing Native Clustering: Start and Stop a Server in a Multiserver Domain

Note:

On Windows, do not stop the Oracle Event Processing server by clicking the **Close** button in the command prompt in which you started it. Always stop the Oracle Event Processing server using the `stopwlevs.cmd` script or `Ctrl-C`.

Deploy Applications to Servers

Once you have created and configured an Oracle Event Processing server and domain, you can deploy Oracle Event Processing applications to them. See:

- Standalone-Server Domain Application Deployment
- Target Server Groups

Manage Applications, Servers, and Domains

Once you have deployed applications to an Oracle Event Processing server and domain, you must manage the application to perform tasks such as monitor its performance and perform upgrades. See:

- Server Administration Tools
- Manage with JMX

Part II

Standalone-Server Domains

Standalone-Server Domains contains the following chapters:

- Standalone-Server Domains
- Standalone-Server Domain Application Deployment

2

Standalone-Server Domains

An Oracle Event Processing standalone-server domain contains a single Oracle Event Processing server. By default, the Configuration Wizard creates a standalone-server domain, which can be the starting point for a multiserver domain.

This chapter includes the following sections:

2.1 Configuration Wizard

The Configuration Wizard is an administration tool that enables you to create a new domain or update an existing domain. You can use the Configuration Wizard in graphical mode (interactive) or silent mode.

Silent mode is non-interactive and requires an XML properties file for selecting configuration options. You can run silent-mode configuration as part of a script or from the command line. The advantage to silent-mode configuration is that if you plan to create a multiserver domain, you can set the domain configuration options once and use the same options to set the configuration on the other servers.

The following procedures show how to perform the create and update operations in graphical mode and in silent mode. You can update only the listen port and the JDBC data source configuration of a standalone-server domain.

2.2 Create a Standalone-Server Domain

Use the Configuration Wizard to create a new domain to which to deploy your applications. The Configuration Wizard creates a single default server in the domain. All of the server files are in a subdirectory of the domain directory. The domain directory has the same name as the server. You also configure the following:

- Server administration user name and password.
- A database or database driver that is different from the default.
- Server listen port.
- Password for the identity `keystore` and `private keystore`.

2.2.1 Create a Standalone-Server Domain in Graphical Mode

Once launched, the Configuration Wizard in graphical mode is self-explanatory, but the full procedure is provided here for your information.

1. Go to the */Oracle/Middleware/my_oep/*`oep/common/bin` directory.
2. Run the `config` command to start the wizard:

   ```
   UNIX:
   ./config.sh

   Windows: config.cmd
   ```

   The Oracle Welcome screen displays.
3. Click **Next**.

   The Choose Create or Update Domain screen displays.
4. In the Choose Create or Update Domain window, select **Create a New OEP Domain** and click **Next**.

   The Configure Administrator Username and Password screen displays.
5. In the Configure Administrator Username and Password screen, enter the following:

   User name: **oepadmin** User password: **welcome1** Confirm user password: **welcome1**
6. Click **Next**.
7. Enter basic configuration information about the default server in the domain. In particular:

   The Configure Server screen displays.

   - Enter the name of the default server. This name is also used for the name of the directory that contains the default server files.
   - The listen port for Oracle Event Processing itself. Default is 9002.
8. In the Configure Server screen, enter the server name and listen port as follows:

   Server name: The name of the default server. This name is also the name of the directory that contains the default server files.

   Server listen port: The port where Oracle Event Processing listens for events. The default is 9002.
9. Click **Next**.

   The Configure Domain Identity Keystore screen displays.
10. In the Configure Domain Identity Keystore screen, enter the following:

    Keystore file: Accept the default. Keystore password: **welcome1**. Confirm keystore password: **welcome1**.

    By default, the password for the certificate private key is the same as the password for the domain identity keystore.

11. Click **Next**.

    The Configuration Options screen displays.

12. In the Configuration Options screen, decide whether or not to update the JDBC data source configuration.

    If you select No, no JDBC data source is configured. If select Yes, you proceed to the page in which you can enter the JDBC data source information.

    a. To create a JDBC data source:

    - Select **Yes** and Click **Next**.

      The Configure Database Properties screen displays.

    - In the Configure Database Properties screen, enter the new JDBC data source values.

      In the top section, enter the data source name, type, driver name and location.

      In the lower section, enter information about the data base to which this data source connects.The JDBC connection URL is generated based on the information you enter.

    - Click **Next**.

      The Create OEP Domain screen displays

    b. To not create a JDBC data source, select **No** and click **Next**.

       The Create OEP Domain screen displays.

13. In the Create OEP Domain screen, enter the following information:

    Domain name: The name of the new domain. Domain location: Accept the default.

    The Configuration Wizard creates the domain in the domain location directory with the domain name that you specified

14. Click **Create**.

    The Creating Domain screen displays.

    If the creation of the domain succeeds, a message similar to the following displays in the information window:

    ```
    Domain created successfully!
    Domain location: C:\Oracle\Middleware\my_oep\user_projects\domains\oep_domain
    ```

15. Click **Done** to exit Configuration Wizard.

16. Go to the domain location to see the domain you just created.

2.2.2 Create a Standalone-Server Domain in Silent Mode

This section describes the procedure to create a standalone-server domain in silent mode.

Silent mode is a non-interactive way to update a domain and requires an XML properties file for selecting configuration options.

- Create an XML Properties File

- Use Silent Mode and Generate a Log File
- Return Exit Codes to the Command Window

2.2.2.1 Create an XML Properties File

1. Go to the computer on which you want to run the Configuration Wizard in silent mode.
2. In an XML editor, create an empty file.

   You can name the file anything you want as long as it has a `.xml` extension. For this procedure, the file name is `silent.xml`.
3. Copy the contents of the following sample XML file into the `silent.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<bea-installer xmlns="http://www.bea.com/plateng/wlevs/config/silent">
  <input-fields>
    <data-value name="CONFIGURATION_OPTION" value="createDomain" />
    <data-value name="USERNAME" value="wlevs" />
    <data-value name="PASSWORD" value="wlevs" />
    <data-value name="SERVER_NAME" value="my_wlevs_server" />
    <data-value name="DOMAIN_NAME" value="myDomain" />
    <data-value name="DOMAIN_LOCATION"
           value="C:\Oracle\Middleware\my_oep\user_projects\domains" />
    <data-value name="NETIO_PORT" value="9002" />
    <data-value name="KEYSTORE_PASSWORD" value="my_keystore_password" />
    <data-value name="PRIVATEKEY_PASSWORD" value="my_privatekey_password" />
    <data-value name="DB_URL" value="jdbc:bea:oracle://localhost:1521:XE" />
    <data-value name="DB_USERNAME" value="db_user" />
    <data-value name="DB_PASSWORD" value="db_password" />
  </input-fields>
</bea-installer>
```

4. In the `silent.xml` file, edit the values for the keywords shown in Table 2-1 to reflect your configuration.

   For example, to create a new domain in the `/Oracle/Middleware/my_oep/finance_projects/domains` directory, update the corresponding `<data-value>` element as follows

```
<data-value name="DOMAIN_LOCATION"
  value="/Oracle/Middleware/my_oep/finance_projects/domains" />
```

Table 2-1 Values for the silent.xml File

For this data-value name...	Enter the following value...
`CONFIGURATION_OPTION`	Whether you want to create a new domain with a default server or update a server in an existing domain. Valid values are `createDomain` or `updateDomain`. Default value is `createDomain`.
`EXISTING_DOMAIN_PATH`	The full path name of an existing server in the domain. Use this option only when updating an existing server in a domain.
`USERNAME`	The user name of the administrator of the created or updated server in the domain.

For this data-value name...	Enter the following value...
PASSWORD	The password of the administrator of the created or updated server in the domain.
SERVER_NAME	The name of the new server in this domain. This name will also be used as the name of the directory that contains the server files.
DOMAIN_NAME	The name of the domain.
DOMAIN_LOCATION	The full name of the directory that will contain the domain. The standard location for Oracle Event Processing domains is */Oracle/Middleware/my_oep/*user_projects/domains.
NETIO_PORT	The port number to which the Oracle Event Processing server instance itself listens.
KEYSTORE_PASSWORD	The password for the Oracle Event Processing identity keystore.
PRIVATEKEY_PASSWORD	The password for the certificate private key. The default value of this option is the value of the KEYSTORE_PASSWORD.
DB_URL	The URL used to connect to a database using JDBC. This option is used to configure the data source. The database configuration parameters are optional; if you do not specify them, then no data source is configured for the server.
DB_USERNAME	The name of the user that connects to the database via the data source. The database configuration parameters are optional; if you do not specify them, then no data source is configured for the server.
DB_PASSWORD	The password of the user that connects to the database via the data source. The database configuration parameters are optional; if you do not specify them, then no data source is configured for the server.

5. Save the file in a directory of your choice.

2.2.2.2 Use Silent Mode and Generate a Log File

1. Go to the */Oracle/Middleware/my_oep/*oep/common/bin directory:
2. Run the `config` command to start the wizard in silent mode and generate a log file to catch failures do to incorrect XML property entries:

```
UNIX:
./config.sh -mode=silent -silent_xml=path_to_xml_file -log=/logs/create_domain.log

Windows: config.cmd -mode=silent -silent_xml=path_to_xml_file -log=/logs/
create_domain.log
```

path_to_xml_file is the full path name to the XML properties file you created

`create_domain.log` specifies to create a log file in the domain directory.

The command does not return any messages if it completes successfully. See Return Exit Codes to the Command Window to get information about the success or failure of the silent execution of the Configuration Wizard.

2.2.2.3 Return Exit Codes to the Command Window

When you run in silent mode, the Configuration Wizard generates exit codes that indicate the success or failure of the creation and configuration of the domain. These exit codes are shown in the following table.

Table 2-2 Exit Codes

Code	Description
0	Configuration Wizard execution completed successfully
-1	Configuration Wizard execution failed due to a fatal error
-2	Configuration Wizard execution failed due to an internal XML parsing error

The following example provides a sample Windows command file that runs the Configuration Wizard in silent mode and echoes the exit codes to the command window from which the script is executed.

```
rem Execute the Configuration Wizard in silent mode
@echo off
config.cmd -mode=silent -silent_xml=c:\scripts\silent.xml  -log=C:\logs
\create_domain.logs

@rem Return an exit code to indicate success or failure
set exit_code=%ERRORLEVEL%

@echo.
@echo Exitcode=%exit_code%
@echo.
@echo Exit Code Key
@echo ---------------
@echo  0=Configuration Wizard completed successfully
@echo -1=Configuration Wizard failed due to a fatal error
@echo -2=Configuration Wizard failed due to an internal XML parsing error
@echo.
```

2.3 Update a Standalone-Server Domain

Use the Configuration Wizard to update an existing standalone-server domain. You can update only the listen port and the JDBC data source configuration.

2.3.1 Update a Standalone-Server Domain in Graphical Mode

Once launched, the Configuration Wizard in graphical mode is self-explanatory, but the full procedure is provided here for your information.

1. Go to the */Oracle/Middleware/my_oep/oep/common/bin* directory.
2. Run the `config` command to start the wizard:

```
UNIX:
./config.sh
```

```
Windows: config.cmd
```

The Oracle Welcome screen displays.

3. Click **Next**.

   The Choose Create or Update Domain screen displays.

4. In the Choose Create or Update Domain window, select **Updating an existing Oracle Event Processing domain**.

5. Click **Next**.

   The Choose an Existing OEP domain screen displays.

6. In the Choose an Existing OEP domain screen text box, enter or browse for the full path name to the server directory for the server that you want to update.

   In this example, the value is `C:\Oracle\Middleware\my_oep \user_projects\domains\myDomain\productionServer`.

7. Click **Next**.

   The Choose an Existing OEP Domain screen displays.

8. In the Choose an Existing OEP Domain screen, select the path to the domain that you want to update from the drop-down list and click `Next`.

   The Configure Server screen displays. The Server name field is grayed out, but you can change the value in the Server listen port field.

   Note:

   To prevent any conflicts when all servers are running at the same time, be sure that you do not enter the same values used by other servers in the domain.

   ---

9. In the Configure Server screen, either change the listen port and click **Next** or click **Next** without changing the listen port.

   The Configuration Options screen displays.

10. In the Configuration Options screen, decide whether to update the JDBC data source configuration.

    To update the JDBC data source configuration:

    a. In the Configuration Options screen, select **Yes** and click **Next**.

       The Configure Database Properties screen displays.

    b. In the Configure Database Properties screen, enter your changes and click **Next**.

       The Create OEP Domain screen displays.

    To leave the JDBC data source configuration unchanged:

    a. In the Configuration Options screen, select **No**.

    b. Click **Next**.

       The Create OEP Domain screen displays.

11. In the Create OEP Domain screen, click **Update** to update the server.

2.3.2 Update a Standalone-Server Domain in Silent Mode

Silent mode is a non-interactive way to update a domain and requires an XML properties file for selecting configuration options.

1. Create an XML properties file that contains the updates that you want to make.

   See Return Exit Codes to the Command Window if you need information about how to create the file, but handle the update settings as follows:

   - Set `CONFIGURATION_OPTION` to `updateDomain`.
   - Set `EXISTING_DOMAIN_PATH` to the full path name of the server directory that contains the server files that you want to update.
   - Do *not* set the `DOMAIN_NAME` and `DOMAIN_LOCATION` options. This is because the Configuration Wizard already knows these values, based on what you entered for `EXISTING_DOMAIN_PATH`.
   - Set the listen port to the new values.Be sure that the new server configuration options, such as `NETIO_PORT` are different than the options for any other servers in the domain.
   - Set the JDBC data source options to the new values. The database options can be the same if you want the updated server to connect to the same database as the other servers.

   The following XML properties file updates the listen port (`NETIO_PORT`) and data source settings for `\~mydomain\productionServer`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<bea-installer xmlns="http://www.bea.com/plateng/wlevs/config/silent">
  <input-fields>
    <data-value name="CONFIGURATION_OPTION" value="updateDomain" />
    <data-value name="EXISTING_DOMAIN_PATH"
      value="C:\Oracle\Middleware\my_oep\user_projects\domains\myDomain
\productionServer" />
    <data-value name="NETIO_PORT" value="9102" />
    <data-value name="DB_URL" value="jdbc:bea:oracle://localhost:1521:XE" />
    <data-value name="DB_USERNAME" value="db_user" />
    <data-value name="DB_PASSWORD" value="db_password" />
  </input-fields>
</bea-installer>
```

2. Go to the */Oracle/Middleware/my_oep/*`oep/common/bin` directory.

3. Run the `config` command to start the wizard in silent mode and generate a log file:

```
UNIX:
./config.sh -mode=silent -silent_xml=path_to_xml_file -log=/logs/create_domain.log

Windows: config.cmd -mode=silent -silent_xml=path_to_xml_file -log=/logs/
create_domain.log
```

   `path_to_xml_file` is the full path name to the XML properties file you created

   `create_domain.log` specifies to create a log file in the domain directory.

The command does not return any messages if it completes successfully. See Return Exit Codes to the Command Window to get information about the success or failure of the silent execution of the Configuration Wizard.

2.4 Start and Stop a Server in a Standalone-Server Domain

You can start and stop an Oracle Event Processing standalone-server with any of the command-line scripts or with Oracle Event Processing Visualizer. For information about Visualizer, see *Using Visualizer for Oracle Stream Explorer*.

- Start a Standalone-Server with the startwlevs Script
- Stop a Standalone-Server with the stopwlevs Script.

2.4.1 Start a Standalone-Server with the startwlevs Script

Each Oracle Event Processing server directory contains a command script that starts a server instance. By default, the script is called `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX).

1. Ensure that the `JAVA_HOME` variable in the server start script points to the correct JDK. If it does not, edit the script.

   The server start script is located in the server directory under the main domain directory. For example, the default server directory for the HelloWorld domain is in */Oracle/Middleware/my_oep/oep/examples/domains/helloworld_domain/defaultserver*.

2. Open a command window and change to the server directory of the domain directory. For example, to start the HelloWorld sample server:

   ```
   cd C:\Oracle\Middleware\my_oep\oep\examples\domains\helloworld_domain
   \defaultserver
   ```

   Note:

   You must run the start scripts from within the target directory. Oracle Event Processing does not support relative directory paths.

3. Run the `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX) script:

   ```
   startwlevs.cmd
   ```

   Note:

   On HP-UX, to avoid an `OutOfMemoryError`, you might need to increase the `MaxPermSize` to 256 in `startwlevs.sh`. For example: `-XX:MaxPermSize=256m`.

2.4.2 Stop a Standalone-Server with the stopwlevs Script

Each Oracle Event Processing server directory contains a command script that stops a server instance; by default, the script is called `stopwlevs.cmd` (Windows) or `stopwlevs.sh` (UNIX).

Note:

The following procedure does not stop an Oracle Event Processing stand-alone server running in SSL mode. To stop an Oracle Event Processing stand-alone server running in SSL mode, run the wlevs.Admin utility, as described in Run wlevs.Admin Utility in SSL Mode.

1. Open a command window and change to the server directory. For example, to stop the running HelloWorld sample server:

   ```
   cd C:\Oracle\Middleware\my_oep\oep\examples\domains\helloworld_domain
   \defaultserver
   ```

2. Execute the `stopwlevs.cmd` (Windows) or `stopwlevs.sh` (UNIX) script.

   Use the `-url` argument to pass the URL that establishes a JMX connection to the server you want to stop. This URL takes the form `service:jmx:msarmi://`*`host`*`:`*`port`*`//jndi/jmxconnector`, where *`host`* refers to the computer hosting the server and *`port`* refers to the server's JNDI port, configured in `config.xml` file. For example:

   ```
   stopwlevs.sh -url service:jmx:msarmi://ariel:9002/jndi/jmxconnector
   ```

   In the example, the host is `ariel` and the JMX port is `9002`. The 9002 port is the `netio` port defined in the Oracle Event Processing server config.xml configuration file. MSA security uses it for JMX connectivity.

   See Connection Arguments for additional details about the `-url` argument.

   Note:

   On Windows, do not stop the Oracle Event Processing server by clicking the **Close** button in the command prompt in which you started it. Always stop the Oracle Event Processing server with the `stopwlevs.cmd` script or with `Ctrl-C`.

3

Standalone-Server Domain Application Deployment

To deploy an application to a standalone-server domain, use either Oracle Event Processing Visualizer or the Deployer utility. This chapter describes how to use the Deployer utility.

See *Using Visualizer for Oracle Event Processing* for information about using Visualizer to deploy an application.

3.1 Deploy with the Deployer Utility

A standalone-server domain has one server (a singleton). When you deploy an Oracle Event Processing application to the singleton server with the Deployer utility, you can specify a server group to which to deploy. Whether or not you specify a server group, Oracle Event Processing deploys the application to the single server in the standalone-server domain. In a multiserver domain, you can organize the servers into server groups and deploy applications specific server groups. See Server Groups.

The following example shows how to deploy to a singleton server group. The command does not include the `-group` option for deployment to a singleton server group:

```
java -jar wlevsdeploy.jar -url http://ariel:9002/wlevsdeployer -install myapp_1.0.jar
```

In the example, the `myapp_1.0.jar` application is deployed to the singleton server group that contains a single server, which is the server running on host `ariel` and listening to port `9002`.

For more information about the Deployer utility, see Deployer Command-Line Reference.

Part III

Multiserver Domains

Multiserver Domains contains the following chapters:

- About Multiserver Domains
- Multiserver Domains with Oracle Coherence
- Multiserver Domains with Native Clustering
- Multiserver Domain Application Deployment

4

About Multiserver Domains

Oracle Event Processing multiserver domains (clusters) are created with Oracle Coherence or Oracle Event Processing native technology. You add one or more servers to a multiserver domain that become logically and physically connected by User Datagram Protocol (UDP). All servers in an Oracle Event Processing multiserver domain are aware of all other servers in the domain and any one server can be the access point for changes to the deployments within the domain

This chapter includes the following sections:

4.1 Multiserver Administration

You administer servers in a multiserver domain at the domain (infrastructure) level. You must configure every server in the domain with the same multicast address, multicast port number, and the same domain name to avoid configuration errors. For example, Oracle Event Processing raises an error when you configure servers with the same multicast address and multicast port but with different domain names.

Every server in the domain detects failures, starts, and restarts by the other servers. You can deploy an application to one server and undeploy it from another server under the same domain.

4.1.1 Oracle Coherence

When you create a multiserver domain with Oracle Coherence, the domain receives replicated and distributed (partitioned) data management services on top of a reliable and highly scalable peer-to-peer clustering protocol. Oracle Coherence has no single points of failure, but instead transparently fails over and redistributes its clustered data management services when a server becomes inoperative or disconnected from the network. When you add a new server or restart a failed server, the server joins the cluster and Oracle Coherence transparently restores services and redistributes the cluster load.

Note:

To use Oracle Event Processing with Oracle Coherence, you must first obtain a valid Oracle Coherence license. See `http://www.oracle.com/technetwork/middleware/coherence/overview/index.html`.

4.1.2 Oracle Event Processing Native Clustering

When you create a multiserver domain with Oracle Event Processing native clustering, the domain receives a native clustering implementation based on TOTEM. However, with Oracle Event Processing native clustering, you cannot take advantage of Oracle Event Processing high availability quality of service options. See Multiserver Domains with Native Clustering.

4.2 Server Groups

A server group is a set of one or more servers with a unique name within the domain. In an Oracle Event Processing domain, an arbitrary number of server groups can exist with a configurable server group membership. A server can be a member of more than one server group. Users are not aware of the underlying server groups because server groups serve as an administration tool that enables you to deploy and manage a multiserver domain at a finer level.

When you deploy applications to the default server group in a multiserver domain, they are deployed to all servers in the domain. All servers in the multiserver domain must have the same correct configuration resources that are required by the application

Oracle Event Processing provides the following predefined deployment server groups:

- Singleton Server Deployment Group
- Domain Deployment Group

You can also create a custom deployment server group so that you can deploy applications to specific servers within a multiserver domain. See Custom Deployment Groups. If you plan to deploy an Oracle Event Processing high availability application and require scalability, you might need to configure an Oracle Event Processing high availability notification group. See *Using Visualizer for Oracle Event Processing*.

You create a server group by deciding on a name and using that name it in the `groups` element in the `config.xml` file for the servers you want to include in the server group.

4.2.1 Singleton Server Deployment Group

The singleton server deployment group consists of one local server only. The membership of this server group depends on the server from which it is accessed. You can use this server group to pin deployments to a single server.

For more information, see Multiserver Domains with Oracle Coherence.

4.2.2 Domain Deployment Group

The domain deployment group contains all live members of the domain. Its membership can only be changed by an administrator.

The domain name is determined by the Oracle Event Processing server `config.xml` file `domain` element. The default domain name is `AllDomainMembers`. In the following example, the default domain name is changed to `myDomain` in the following `config.xml` file entry:

```
<domain>
    <name>myDomain</name>
</domain>
```

4.2.3 Custom Deployment Groups

There are cases where the application logic cannot be replicated across a homogenous set of servers in a multiserver domain. An example is an application that determines the best price provided by different pricing engines. Another example is an application that sends an alert when a position crosses a threshold. In either case, the application does not perform multiple operations, but instead calculates once or sends a single event, respectively. In other cases, the application has a singleton nature, such as monitoring an application, the HTTP pub-sub server, and so on.

A more complex example is a domain with two applications. The first application, `strategies`, uses several strategies to calculate different prices for a derivative and feeds the results to the `selector` application. The `selector` application selects the best price from the results sent to it by the `strategies` application.

The `strategies` application can be replicated to achieve fault-tolerance. However, the `selector` application must keep state so it can determine the best price. Because `selector` must maintain state, the `selector` application *cannot* be replicated across a homogenous set of servers.

If a domain must support servers that are not completely homogeneous, you can configure with custom deployment groups. Applications deployed to a custom deployment group in a multiserver domain are deployed homogeneously to all servers within the deployment group. All servers within the deployment group must have the appropriate configuration resources required by the application or applications.

For more information, see Multiserver Domains with Oracle Coherence.

4.3 Multiserver Notifications and Messaging

Oracle Event Processing provides a number of notification and messaging APIs for server groups and servers. You can use these APIs to configure a server to receive notification when its server group or domain membership changes. The change can be because an administrator changed it or because of a server failure. You can also use these APIs to send messages to individual server groups and to the domain.

When you configure your application to use Oracle Event Processing high availability options, the primary Oracle Event Processing server uses Oracle Coherence to communicate with its secondary servers to keep them up-to-date with the event processing progress of the primary server.

You can also configure Oracle Event Processing servers in a multiserver domain to communicate securely.

For more information, see:

- Secure the Messages Sent Between Servers
- Secure the Messages Sent Between Servers in a Multiserver Domain.

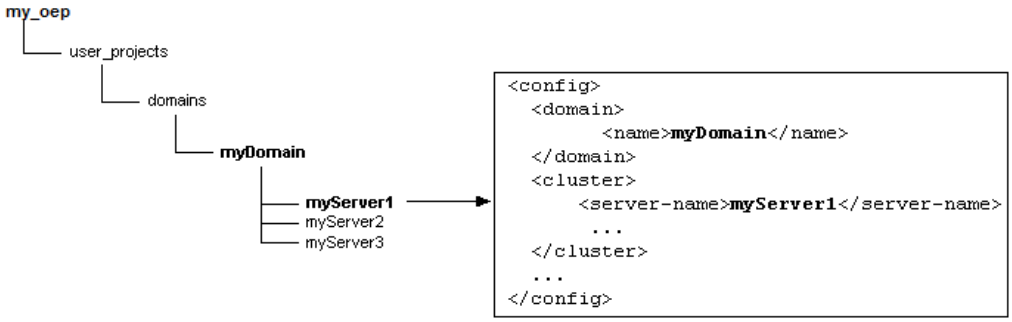
4.4 Multiserver Domain Directory Structure

Servers in an Oracle Event Processing domain store their files in a single directory. By convention, the directories of the servers in a multiserver domain are subdirectories of the domain directory. Also, the name of the servers and domain correspond to the name of the server directories and domain directory, respectively.

This is by convention only, and not required, although Oracle recommends you set up your domains this way for simplicity and consistency. If the servers of the multiserver domain are located on different computers, you can replicate the directory structure on both computers, also for simplicity and consistency.

Figure 4-1 shows a multiserver domain directory with three servers.

Figure 4-1 Multiserver Domain Directory Structure



In Figure 4-1, the `myServer1` configuration file snippet shows how the domain directory and domain object are configured with the same name, and the server directory and server name.

The domain directory is in the */Oracle/Middleware/my_oep/*`user_projects/domains` directory, which is the default location for Oracle Event Processing domains.

4.5 Order of Cluster Element Child Elements

The order of `cluster` element child elements in the `config.xml` file is important. If you put elements in an incorrect order you can get an error. The following list describes the order in which to list the child elements:

- `server-name`
- `server-host-name`: Specifies the host address/IP used for point-to-point HTTP multiserver communication. Default value is `localhost`.

  This element is mandatory if one or more Oracle Event Processing servers in your multiserver domain are on different hosts and you plan to manage the multiserver domain with Oracle Event Processing Visualizer. The element is also mandatory when a server is deployed on a host machine that has multiple IP addresses configured (whether in a multiserver or standalone-server environment).
- `multicast-address`: The multicast communication address. For Oracle Coherence well-known addressing (WKA) a unicast address can be used.
- `multicast-port`: Optional. Specifies the port to use for multicast traffic. Default value is `9001`.

- `identity`: Mandatory only for Oracle Event Processing native clustering. This element is not used for Oracle Coherence.
- `enabled`
- `security`
- `groups`
- `operation-timeout`: Optional. Specifies, in milliseconds, the time out for point-to-point HTTP multiserver requests. Default value is `30000`.

For a complete description of the Oracle Event Processing server `config.xml` file `cluster` element, see *Using Visualizer for Oracle Event Processing*.

4.6 High Availability and Multiserver Domains

If you use Oracle Coherence clustering for your multiserver domain, you can take advantage of Oracle Event Processing high availability quality of service options. These options are not supported by Oracle Event Processing native clustering. For more information, see *Schema Reference for Oracle Event Processing*.

4.7 Scalability and Multiserver Domains

With either Oracle Coherence or Oracle Event Processing native clustering, you can take advantage of Oracle Event Processing scalability quality of service options. For more information, see *Schema Reference for Oracle Event Processing*.

5

Multiserver Domains with Oracle Coherence

You can create, configure, and administer multiserver domains that are based on Oracle Coherence. With Oracle Coherence, the domain receives replicated and distributed (partitioned) data management services on top of a reliable and highly scalable peer-to-peer clustering protocol.

This chapter includes the following sections:

If you plan to deploy an Oracle Event Processing high availability application and require scalability, you might need to create an Oracle Event Processing high availability notification group. For more information, see *Schema Reference for Oracle Event Processing*.

5.1 Create a Multiserver Domain

To create a multiserver domain, start by using the Configuration Wizard to create one domain with one server. This is a standalone-server domain. To convert the standalone-server domain to a multiserver domain, you can either:

- Copy and rename the server you just created and then edit the `config.xml` file for each server so that they all have the same multicast name, multicast port, and domain name.
- Use the Configuration Wizard to generate additional stand-alone servers and then edit the `config.xml` file for each server so that they all have the same multicast name, multicast port, and domain name.

The procedures in this chapter use the first approach.

5.2 Create a Multiserver Domain with Default Groups

You can create a multiserver domain that uses only the two predefined deployment groups described in Singleton Server Deployment Group. The predefined deployment

groups are the singleton group and the domain group. In a domain that uses default groups, all servers must be completely homogeneous.

If a domain must support servers that are not completely homogeneous, configure this by creating custom server groups. See Create a Multiserver Domain with Custom Groups.

Create a Multiserver Domain with Default Server Groups

1. Use the Configuration Wizard to create a standalone-server domain.

   See Standalone-Server Domains.

2. Change to the directory where you put the new standalone-server domain when you created it.

   By default, the location is

   `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/`

3. Copy and rename the server directory until you have the number of servers that you need under the domain.

4. Update the `config.xml` file for each member server by adding a `cluster` child element of the root `config` element. The following example shows a server that is part of a domain called `myDomain`.

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer1</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <enabled>true</enabled>
  </cluster>
...
</config>
```

   You can use the following child elements of cluster:

   - `server-name`: The `server-name` child element of `cluster` specifies a unique name for the server. Oracle Event Processing Visualizer *Using Visualizer for Oracle Event Processing* uses the value of this element when it displays the server in its console. The default is the Oracle Coherence member name if that is set or `WLEvServer-`*`MEMBERID`*.

   - `server-host-name`: Specifies the host address/IP used for point-to-point HTTP multiserver communication. Default value is `localhost`. This element is mandatory if one or more Oracle Event Processing servers in your multiserver domain are on different hosts and you plan to manage the multiserver domain using Oracle Event Processing Visualizer. It is also mandatory if a server is deployed on a host machine that has multiple IP addresses configured (whether in a multiserver or standalone-server environment).

   - `multicast-address`: The `multicast-address` element is required unless all servers of the multiserver domain are hosted on the same computer; in that case you can omit the `multicast-address` element and Oracle Event Processing automatically assigns a multicast address to the multiserver domain based on the computer's IP address. If the servers are hosted on different

computers, then you must provide an appropriate domain-local address. Oracle recommends that you use an address of the form `239.255.X.X`, which is what the auto-assigned multicast address is based on. All the Oracle Event Processing servers using this `multicast-address` must be on the same subnet. With Oracle Coherence, you can specify a unicast address here and Oracle Coherence will use WKA (Well Known Addressing).

- `enabled`: By default the clustering of the servers in a multiserver domain is disabled for Oracle Coherence. The element `<enabled>true</enabled>` is required if you want to enable Oracle Coherence.

For each server of the multiserver domain, the `multicast-address` elements must contain the same value. The `server-name` element must be different for each server in the multiserver domain. The following example shows the `config.xml` file of a second server, called `myServer2`, in the `myDomain` multiserver domain.

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>MyServer2</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <enabled>true</enabled>
  </cluster>
...
</config>
```

See Secure the Messages Sent Between Servers for a description of additional multiserver-related configuration elements and the required order of child elements.

5. Optionally, override the default Oracle Coherence clustering configuration on all of the servers in the multiserver domain, if necessary.

   See Configure the Oracle Coherence Cluster.

6. Optionally, secure the messages that are shared among all of the servers in the multiserver domain by configuring encryption and digital signatures.

   See Secure the Messages Sent Between Servers.

7. Consider enabling Oracle Event Processing Visualizer on a subset of machines within the multiserver domain so that in the event of a failure, you can use Visualizer to troubleshoot from a machine that is still operating.

   Note:

   Enabling Oracle Event Processing Visualizer on a given Oracle Event Processing Server can impact the performance of the server depending on the Oracle Event Processing Visualizer workload.

8. Start all of the servers in your multiserver domain.

   See Start and Stop a Server in a Multiserver Domain.

5.3 Create a Multiserver Domain with Custom Groups

Use this procedure if you plan to deploy applications that are not homogeneous and require a custom configuration. If all of the servers in your domain are completely homogeneous, you do not need to create custom server groups. Instead, use the predefined default server groups: the singleton group and the domain group. See Create a Multiserver Domain with Default Groups.

In this procedure, assume you have created three servers: `myServer1`, `myServer2`, and `myServer3`. You want `myServer1` to be a member of the `selector` server group and `myServer2` and `myServer3` to be members of the `strategy` server group.

Create a Multiserver Domain with Custom Server Groups

1. Use the Configuration Wizard to create a standalone-server domain.

   See Standalone-Server Domains.

2. Copy and rename the server directory until you have the number of servers that you want under the domain.Update the `config.xml` file for each member server by adding a `cluster` child element of the root `config` element. Within the `cluster` child element, add a `group` element to specify the server group.

   Note:

   When you add `cluster` element child elements, observe the correct element order as Order of Cluster Element Child Elements describes.

   The `groups` element can include more than one server group name in a case where the server is a member of more than one server group. Separate multiple server group names with commas.

   The `groups` element is optional. If a server configuration does not have a `groups` element, then the server is a member of the default server groups (domain and singleton). For more information about the domain and singleton groups, see Server Groups.

   The following examples show the relevant snippets from the `config.xml` file for `myServer1`, `myServer2`, and `myServer3`. The `groups` entry for `myServer1` puts it in the `selector` group. The `groups` entry for `myServer2` and `myServer3`, put them in the `strategy` group.

   myServer1:

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer1</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <enabled>true</enabled>
      <groups>selector</groups>
  </cluster>
...
</config>
```

myServer2:

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer2</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <enabled>true</enabled>
      <groups>strategy</groups>
  </cluster>
...
</config>
```

myServer3:

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer3</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <enabled>true</enabled>
      <groups>strategy</groups>
  </cluster>
...
</config>
```

3. Optionally, override the default Oracle Coherence clustering configuration on all of the servers in the multiserver domain, if necessary.

   See Configure the Oracle Coherence Cluster.

4. Optionally, secure the messages that are shared among all of the servers in the multiserver domain by configuring encryption and digital signatures.

   See Secure the Messages Sent Between Servers.

5. Consider enabling Oracle Event Processing Visualizer on a subset of machines within the multiserver domain so that in the event of a failure, you can use Visualizer to troubleshoot from a machine that is still operating.

   Note:

   Enabling Oracle Event Processing Visualizer on a given Oracle Event Processing Server can impact the performance of the server depending on the Oracle Event Processing Visualizer workload.

6. Start all of the servers in your multiserver domain.

   See Start and Stop a Server in a Multiserver Domain.

5.4 Configure the Oracle Coherence Cluster

For Oracle Coherence cluster configuration, Oracle Event Processing uses the Oracle Coherence `tangosol-coherence-override.xml` configuration file. The `tangosol-coherence-override.xml` file is a global per-server configuration file.

It is referred to as operational configuration in the Oracle Coherence documentation. Put this file with the `config.xml` server configuration file, which is in the Oracle Event Processing server `config` directory.

When you declare that a caching system uses the Oracle Coherence provider, make sure that all of the caches of the caching system map to an Oracle Coherence configuration and not to an Oracle Event Processing local configuration. If you do not do this and you have one or more caches mapping to an Oracle Event Processing local configuration, Oracle Event Processing throws an exception.

Servers in a multiserver domain must be configured with the same multicast address and port number and the same domain name. For example, Oracle Event Processing throws an error if you configure servers with the same multicast address and port number, but with different domain names.

The `tangosol-coherence-override.xml` file supports the following elements: `<cluster-config>`, `<management-config>`, and `<logging-config>`. You cannot override the cluster name because Oracle Event Processing always sets the cluster name to the domain name. Choose a unique name for each Oracle Event Processing domain to prevent accidental cluster discovery between different domains.

The following sample shows a simple configuration that specifies the `time-to-live` setting that determines the maximum number of hops a packet can traverse. A hop is measured as a traversal from one network segment to another via a router.

```
<?xml version='1.0'?>
<coherence xml-override="/tangosol-coherence-override.xml">
  <cluster-config>
    <multicast-listener>
      <time-to-live>3</time-to-live>
    </multicast-listener>
    ...
</coherence>
```

For detailed information about the `tangosol-coherence-override.xml` file, see Oracle Coherence Developer's Guide.

5.5 Update a Multiserver Domain

You update servers in a multiserver domain the same way that you update the one server in a standalone-server domain: You either launch the Configuration Wizard or edit the XML properties file to use silent mode. The only difference between updating one server and multiple servers is that in a multiserver domain, you perform the update on each server individually. Using an XML properties file in silent mode might be your best option in a multiserver domain.

When you use the Configuration Wizard to update a server, you can update only the listen port and the JDBC data source configuration. In silent mode, you can add a server and update anything for which there is a data value.

To update the existing configuration of an existing server you provide values for the following data values in the XML properties file:

- Set `CONFIGURATION_OPTION` to `updateDomain`.
- Set the `DOMAIN_NAME` and `DOMAIN_LOCATION` options to the name and location of the *existing* domain. In our example, the values are `myDomain` and `C:\Oracle\Middleware\my_oep\user_projects\domains`, respectively.
- Set the `SERVER_NAME` option to the name of the *new* server you want to add to the existing domain. In our example, this would be `myServer2`.

- If this server is running on the same computer as the other servers in the multiserver domain, then be sure that the new server configuration options, such as `NETIO_PORT` are different than the options for any existing server in the domain. The database options can be the same if you want the new server to connect to the same database as the existing servers.

  If the server is on a different machine than the other servers in the multiserver domain, then the ports do not have to be different.

The following example is an XML properties file that updates an existing server in a multiserver domain.

```
<?xml version="1.0" encoding="UTF-8"?>
<bea-installer xmlns="http://www.bea.com/plateng/wlevs/config/silent">
  <input-fields>
    <data-value name="CONFIGURATION_OPTION" value="updateDomain" />
    <data-value name="USERNAME" value="wlevs" />
    <data-value name="PASSWORD" value="wlevs" />
    <data-value name="SERVER_NAME" value="myServer2" />
    <data-value name="DOMAIN_NAME" value="myDomain" />
    <data-value name="DOMAIN_LOCATION" value="C:\Oracle\Middleware\my_oep\user_projects
\domains" />
    <data-value name="NETIO_PORT" value="9102" />
    <data-value name="KEYSTORE_PASSWORD" value="my_keystore_password" />
    <data-value name="PRIVATEKEY_PASSWORD" value="my_privatekey_password" />
    <data-value name="DB_URL" value="jdbc:bea:oracle://localhost:1521:XE" />
    <data-value name="DB_USERNAME" value="db_user" />
    <data-value name="DB_PASSWORD" value="db_password" />
  </input-fields>
</bea-installer>
```

Note:

After you create the servlet and if the two servlets are on the same machine, modify the SSL port to recognize the new servlet.

5.6 Secure the Messages Sent Between Servers

The servers in a multiserver domain update their state by exchanging multiserver-related messages. These messages should be checked for integrity. You can use a private key to ensure integrity. The private key must be shared by all of the servers within the domain.

When you use the Oracle Coherence clustering implementation, you can secure the messages sent between servers in a multiserver domain.

Secure Messages Sent Between Servers

1. Stop all servers in your multiserver domain, if they are currently running.

   See Start and Stop a Server in a Multiserver Domain.

2. Edit the `config.xml` file of each server in the multiserver domain by adding the `security` child element to the `cluster` element:

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer1</server-name>
```

```
      <multicast-address>239.255.0.1</multicast-address>
      <identity>1</identity>
      <enabled>coherence</enabled>
      <security>encrypt</security>
  </cluster>
...
</config>
```

By default the `config.xml` file is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config`.

You must specify one of the following values for the `security` child element:

- `none`: No security is configured for the multiserver domain. This is the default value.
- `encrypt`: Encrypt multiserver messages.

Observe the correct order of child elements in the cluster element. See Order of Cluster Element Child Elements.

3. Edit the `Oracle security-config.xml` file for each server in the multiserver domain by adding the `encryption-service` child element to the `config` root element, as shown.

   By default the `security-config.xml` files are in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config/`

   ```
   <config>
     <encryption-service>
        <signature-enabled>true</signature-enabled>
     </encryption-service>
     <css-realm>
      ...
   </config>
   ```

4. Ensure that the *<domainname/<servername>*`/.aesinternal.dat` file for each server in the multiserver domain is exactly the same by copying the file from one server to the other servers.

   This file is created by the Configuration Wizard when you first created the server. Oracle Event Processing uses this file for encrypting messages.

5. Start one of the servers in your domain.

   See Start and Stop a Server in a Multiserver Domain.

   Because of the `encryption-service` element that you added to the `security-config.xml` file, Oracle Event Processing creates the `*.msasig.dat` file in the main server directory. Oracle Event Processing uses this file for digitally signing messages.

6. Stop the server you just started.

   See Start and Stop a Server in a Multiserver Domain.

7. Copy the `*.msasig.dat` file to the other servers.

8. Perform the following steps on each server in the cluster:

- Change to the server directory for your domain.
- Create a keystore `coherence-identity.jks` containing the boot user using the JDK `keytool` utility and the following command line (broken here for readability; in practice the full command should be on one line):

  ```
  keytool -genkey -v -keystore config/coherence-identity.jks
  -storepass PASSWORD -alias BOOT-USER -keypass BOOT-USER-PASSWORD
  -dname CN=BOOT-USER
  ```

  Where:

  – *PASSWORD* is the password used to secure the keystore.

  – *BOOT-USER* is the user name you used to log into the Oracle Event Processing server host.

  – *BOOT-USER-PASSWORD* is the password you used when you logged into the Oracle Event Processing server host.

- Create a `permissions.xml` file.

  By default the `permissions.xml` file goes in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config`

- Edit the `permissions.xml` file to add the following permission for the boot user:

  ```
  <permissions>
      <grant>
          <principal>
              <class>javax.security.auth.x500.X500Principal</class>
              <name>CN=BOOT-USER</name>
          </principal>

          <permission>
              <target>*</target>
              <action>all</action>
          </permission>
      </grant>
  </permissions>
  ```

  Where *BOOT-USER* is the user name you used to log into the Oracle Event Processing server host.

- Save and close the `permissions.xml` file.
- Create a `login.config` file.

  By default the `login.config` file goes in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config/`

- Edit the `login.config` file to add the following:

  ```
  Coherence {
      com.tangosol.security.KeystoreLogin required
          keyStorePath=".${/}config${/}coherence-identity.jks";
  };
  ```

- Save and close the `login.config` file.

- Update the server startup script for your platform, `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX), by adding the following property to the `java` command that actually starts the server:

  ```
  -Djava.security.auth.login.config=./login.config
  ```

  For example (put the full command should be on one line):

  ```
  "%JAVA_HOME%/bin/java" %DGC% %DEBUG%
  -Djava.security.auth.login.config=./login.config
  -Dwlevs.home="%USER_INSTALL_DIR%" -Dbea.hoe="%BEA_HOME%"
  -jar "%USER_INSTALL_DIR%/bin/wlevs.jar" %1 %2 %3 %4 %5 %6
  ```

9. If you plan to use Oracle Event Processing Visualizer with the servers in this domain, see Configure SSL in a Multiserver Domain for Visualizer.

10. Start all servers in your multiserver domain.

    See Start and Stop a Server in a Multiserver Domain.

5.7 Use Multiserver Domain APIs to Manage Group Membership

In an active-active system, you deploy actively executing applications homogeneously across several servers. There are cases when these homogeneously-deployed applications need to elect a coordinator application to lead. In this case, events that result from the coordinator application are kept and passed to the next component in the EPN. The results of secondary servers are dropped. If the coordinator fails, then one of the secondary servers must be elected as the new coordinator.

To enable this behavior in an application, the adapter or event bean, usually in the role of an event sink, must implement the following interface:

`com.bea.wlevs.ede.api.cluster.GroupMembershipListener`

This interface enables the event sink to listen for multiserver domain group membership changes. At runtime, Oracle Event Processing calls the `onMembershipChange` callback method whenever membership changes occur.

The callback method signature follows:

```
onMembershipChange(Server localIdentity, Configuration groupConfiguration);
```

In the implementation of the `onMembershipChange` callback method, the event sink uses the `Server` object (`localIdentity`) to verify if it is the leader. This can be done be comparing `localIdentity` with the result of `Configuration.getCoordinator()` run on the second parameter, `groupConfiguration`. This parameter also allows a server to know what the current members of the group are by executing `Configuration.getMembers()`.

Note:

There is a new API for notification groups. For more information, see *Java API Reference for Oracle Event Processing*.

To only keep events when it is a coordinator, the event sink must get a new `Server` identity every time membership in the group changes. Group membership changes occur when, for example, another server within the group fails and is no longer the coordinator.

The following interface is for listening to membership changes to the domain as a whole, rather than changes to the server group.

`com.bea.wlevs.ede.api.cluster.DomainMembershipListener`

In a hot-hot configuration, there is a non-zero delay in failure notification. If you use the notification APIs to implement clustering, you will lose and not process events that occur in the window of time between the server failure and the notification being delivered to the new master server.

See *Java API Reference for Oracle Event Processing*.

5.8 Start and Stop a Server in a Multiserver Domain

To start the servers in a multiserver domain, start each server separately by running its start script. This is the same way you start a server in a standalone server domain. See Start and Stop a Server in a Standalone-Server Domain for details.

If you have not configured custom groups for the multiserver domain, then all servers are members of the predefined domain group, which contains all of the servers in the multiserver domain, and a singleton group, one for each member server. This means, for example, if there are three servers in the multiserver domain, then there are three singleton groups.

If, however, you have configured custom groups for the multiserver domain, then the servers are members of the groups for which they have been configured and the predefined groups.

Note:

on Windows, do not stop the Oracle Event Processing server by clicking the **Close** button in the command prompt in which you started it. Always stop the Oracle Event Processing server using the `stopwlevs.cmd` script or `Ctrl-C`.

6

Multiserver Domains with Native Clustering

You can create, configure, and administer multiserver domains that are based on native clustering. With native clustering he domain receives a native clustering implementation based on TOTEM, but you cannot take advantage of Oracle Event Processing high availability quality of service options. If you require high availability quality of service options, build your multiserver domain with Oracle Coherence.

This chapter includes the following sections:

6.1 Create a Multiserver Domain

To create a multiserver domain, start by using the Configuration Wizard to create one domain with one server. This is a standalone-server domain. To convert the standalone-server domain to a multiserver domain, you can either:

- Copy and rename the server you just created and then edit the `config.xml` file for each server so that they all have the same multicast name, multicast port, and domain name.
- Use the Configuration Wizard to generate additional stand-alone servers and then edit the `config.xml` file for each server so that they all have the same multicast name, multicast port, and domain name.

The procedures in this chapter use the first approach.

6.2 Create a Multiserver Domain with Default Groups

You can create a multiserver domain that uses only the two predefined deployment groups described in Singleton Server Deployment Group. The predefined deployment groups are the singleton group and the domain group. In a domain that uses default groups, all servers must be completely homogeneous.

If a domain must support servers that are not completely homogeneous, configure this by creating custom groups. See Create a Multiserver Domain with Custom Groups.

Create a multiserver domain with default groups:

1. Use the Configuration Wizard to create a standalone-server domain.

   See Standalone-Server Domains.

2. Change to the directory where you put the new standalone-server domain when you created it.

   By default, the location is

   `Oracle/Middleware/my_oep/user_projects/`<domainname>.

3. Copy and rename the server directory until you have the number of servers that you need under the domain.

4. Update the `config.xml` file for each member server by adding a `cluster` child element of the root config element. The following example shows a server that is part of a domain called `myDomain`.

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer1</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>1</identity>
      <enabled>evs4j</enabled>
  </cluster>
...
</config>
```

   You can use the following child elements of cluster:

   - `server-name`: The `server-name` child element of `cluster` specifies a unique name for the server. Oracle Event Processing Visualizer uses the value of this element when it displays the server in its console. The default value if the element is not set is `Server-`*`identity`* where *`identity`* is the value of the `identity` element.

   - `server-host-name`: Specifies the host address/IP used for point-to-point HTTP multiserver communication. Default value is `localhost`.

     This element is mandatory if one or more Oracle Event Processing servers in your multiserver domain are on different hosts and you plan to manage the multiserver domain using the Oracle Event Processing Visualizer. It is also mandatory if a server is deployed on a host machine that has multiple IP addresses configured (whether in a multiserver or standalone-server environment).

   - `multicast-address`: The `multicast-address` element is required unless all servers of the multiserver domain are hosted on the same computer; in that case you can omit the `multicast-address` element and Oracle Event Processing automatically assigns a multicast address to the multiserver domain based on the computer's IP address.

     If, however, the servers are hosted on different computers, then you must provide an appropriate domain-local address. Oracle recommends you use an

address of the form *239.255.X.X*, which is what the auto-assigned multicast address is based on.

All the Oracle Event Processing servers using this `multicast-address` must be on the same subnet.

- `identity`: The `identity` element identifies the server's identity and must be an integer between 1 and `INT_MAX`. Oracle Event Processing numerically compares the server identities during multiserver operations; the server with the lowest identity becomes the domain coordinator. Be sure that each server in the multiserver domain has a different identity; if servers have the same identity, the results of multiserver operations are unpredictable.
- `enabled`: By default the clustering of the servers in a multiserver domain is disabled for Oracle Coherence, so to enable Oracle Event Processing native clustering use `<enabled>evs4j</enabled>`.

Note:

When adding `cluster` element child elements, observe the correct element order as Order of Cluster Element Child Elements describes.

For each server of the multiserver domain, the `multicast-address` elements must contain the same value. The `identity` and `server-name` elements, however, must be different for each server in the multiserver domain. The following example shows the `config.xml` file of a second server, called `myServer2`, in the `myDomain` multiserver domain. The `identity` value for his server is 2.

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer2</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>2</identity>
      <enabled>evs4j</enabled>
  </cluster>
...
</config>
```

See Order of Cluster Element Child Elements for a description of additional multiserver-related configuration elements and the required order of child elements.

5. Optionally, secure the messages that are shared between the servers in a domain by configuring encryption and digital signatures.

   See Secure the Messages Sent Between Servers in a Multiserver Domain.

6. Consider enabling Oracle Event Processing Visualizer on a subset of machines within the multiserver domain so that in the event of a failure, you can use Visualizer to troubleshoot from a machine that is still operating.

Note:

Enabling Oracle Event Processing Visualizer on a given Oracle Event Processing Server can impact the performance of the server depending on the Oracle Event Processing Visualizer workload.

7. Start all servers in your multiserver domain.

   See Start and Stop a Server in a Multiserver Domain.

6.3 Create a Multiserver Domain with Custom Groups

Use this procedure if you plan to deploy applications that are not homogeneous and require a custom configuration. If all of the servers in your domain are completely homogeneous, you do not need to create custom server groups. Instead, use the predefined default server groups: the singleton group and the domain group. See Create a Multiserver Domain with Default Groups.

In this procedure, assume you have created three servers: `myServer1`, `myServer2`, and `myServer3`. You want `myServer1` to be a member of the `selector` group and `myServer2` and `myServer3` to be members of the `strategy` group.

Create a multiserver domain with Custom Server Groups:

1. Use the Configuration Wizard to create a standalone-server domain.

   See Standalone-Server Domains.

2. Copy and rename the server directory until you have the number of servers that you want under the domain.

3. Update the `config.xml` file for each member server by adding a `cluster` child element of the root `config` element. Within the `cluster` child element, add a `group` element to specify the server group.

   Note:

   When you add `cluster` element child elements, observe the correct element order as Order of Cluster Element Child Elements describes.

   The `groups` element can include more than one server group name in a case where the server is a member of more than one server group. Separate multiple server group names with commas.

   The `groups` element is optional. If a server configuration does have a `groups` element, then the server is a member of the default groups (domain and singleton). For more information about the domain and singleton groups, see Server Groups.

   The following examples show the relevant snippets from the `config.xml` file for `myServer1`, `myServer2`, and `myServer3`. The `groups` entry for `myServer1` puts it in the `selector` group. The `groups` entry for `myServer2` and `myServer3`, put them in the `strategy` group.

   myServer1:

```
<config>
  <domain>
```

```
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer1</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>1</identity>
      <enabled>evs4j</enabled>
      <groups>selector</groups>
  </cluster>
...
</config>
```

myServer2:

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer2</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>2</identity>
      <enabled>evs4j</enabled>
      <groups>strategy</groups>
  </cluster>
...
</config>
```

myServer3:

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer3</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>3</identity>
      <enabled>evs4j</enabled>
      <groups>strategy</groups>
  </cluster>
...
</config>
```

4. Optionally, secure the messages that are shared between the servers in a domain by configuring encryption and digital signatures.

   See Secure the Messages Sent Between Servers in a Multiserver Domain.

5. Consider enabling Oracle Event Processing Visualizer on a subset of machines within the multiserver domain so that in the event of a failure, you can use Visualizer to troubleshoot from a machine that is still operating.

   Note:

   Enabling Oracle Event Processing Visualizer on a given Oracle Event Processing Server can impact the performance of the server depending on the Oracle Event Processing Visualizer workload.

6. Start all servers in your multiserver domain.

See Start and Stop a Server in a Multiserver Domain.

6.4 Update a Multiserver Domain

You update servers in a multiserver domain the same way that you update the one server in a standalone-server domain: You either launch the Configuration Wizard or edit the XML properties file to use silent mode. The only difference between updating one server and multiple servers is that in a multiserver domain, you perform the update on each server individually. Using an XML properties file in silent mode might be your best option in a multiserver domain.

When you use the Configuration Wizard to update a server, you can update only the listen port and the JDBC data source configuration. In silent mode, you can add a server and update anything for which there is a data value.

To update the existing configuration of an existing server you provide values for the following data values in the XML properties file:

- Set `CONFIGURATION_OPTION` to `updateDomain`.
- Set the `DOMAIN_NAME` and `DOMAIN_LOCATION` options to the name and location of the *existing* domain. In the example, the values are `myDomain` and `C:\Oracle\Middleware\my_oep\user_projects\domains`, respectively.
- Set the `SERVER_NAME` option to the name of the *new* server you want to add to the existing domain. In the example, this would be `myServer2`.
- If this server is running on the same computer as the other servers in the multiserver domain, then be sure that the new server configuration options, such as `NETIO_PORT` are different than the options for any existing server in the domain. The database options can be the same if you want the new server to connect to the same database as the existing servers.

  If the server is on a different machine than the other servers in the multiserver domain, then the ports do not have to be different.

The following example is an XML properties file that updates an existing server in a multiserver domain.

```
<?xml version="1.0" encoding="UTF-8"?>
<bea-installer xmlns="http://www.bea.com/plateng/wlevs/config/silent">
  <input-fields>
    <data-value name="CONFIGURATION_OPTION" value="createDomain" />
    <data-value name="USERNAME" value="wlevs" />
    <data-value name="PASSWORD" value="wlevs" />
    <data-value name="SERVER_NAME" value="myServer1" />
    <data-value name="DOMAIN_NAME" value="myDomain" />
    <data-value name="DOMAIN_LOCATION" value="C:\Oracle\Middleware\my_oep\user_projects
\domains" />
    <data-value name="NETIO_PORT" value="9102" />
    <data-value name="KEYSTORE_PASSWORD" value="my_keystore_password" />
    <data-value name="PRIVATEKEY_PASSWORD" value="my_privatekey_password" />
    <data-value name="DB_URL" value="jdbc:bea:oracle://localhost:1521:XE" />
    <data-value name="DB_USERNAME" value="db_user" />
    <data-value name="DB_PASSWORD" value="db_password" />
  </input-fields>
</bea-installer>
```

6.5 Secure the Messages Sent Between Servers in a Multiserver Domain

The servers in a multiserver domain update their state by exchanging multiserver-related messages. These messages should be checked for integrity. You can use a

private key to ensure integrity. The private key must be shared by all of the servers within the domain.

Secure Messages Sent Between Servers

1. Stop all servers in your multiserver domain, if they are currently running.

   See Start and Stop a Server in a Multiserver Domain.

2. Edit the `config.xml` file of each server in the multiserver domain by adding the `security` child element to the `cluster` element.

   By default the `config.xml` file is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config/`

```
<config>
  <domain>
        <name>myDomain</name>
  </domain>
  <cluster>
      <server-name>myServer1</server-name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>1</identity>
      <enabled>evs4j</enabled>
      <security>encrypt</security>
  </cluster>
...
</config>
```

   You must specify one of the following values for the `security` child element:

   - `none`: No security is configured for the multiserver domain. This is the default value.
   - `encrypt`: Encrypt multiserver messages.

   Observe the correct order of child elements in the cluster element. See Order of Cluster Element Child Elements.

3. Edit the `security-config.xml` file for each server in the multiserver domain by adding the `encryption-service` child element to the `config` root element.

   By default the `security-config.xml` file is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config/`

```
<config>
  <encryption-service>
     <signature-enabled>true</signature-enabled>
  </encryption-service>
  <css-realm>
   ...
</config>
```

4. Ensure that the *myDomain/servername/*`.aesinternal.dat` file for each server in the multiserver domain is exactly the same by copying the file from one server to the other servers.

   This file is created by the Configuration Wizard when you first created the server. Oracle Event Processing uses this file for encrypting messages.

5. Start one of the servers in your domain.

   See Start and Stop a Server in a Multiserver Domain.

   Because of the `encryption-service` element that you added to the `security-config.xml` file, Oracle Event Processing creates the `*.msasig.dat` file in the main server directory. Oracle Event Processing uses this file for digitally signing messages.

6. Stop the server you just started.

   See Start and Stop a Server in a Multiserver Domain.

7. Copy the `*.msasig.dat` file to the other servers.

8. If you plan to use Oracle Event Processing Visualizer with the servers in this domain, see Configure SSL in a Multiserver Domain for Visualizer.

9. Start all servers in your multiserver domain.

   See Start and Stop a Server in a Multiserver Domain.

6.6 Use Multiserver Domain APIs to Manage Group Membership Changes

In an active-active system, you deploy actively executing applications homogeneously across several servers. There are cases when these homogeneously-deployed applications need to elect a coordinator application to lead. In this case, events that result from the coordinator application are kept and passed to the next component in the EPN. The results of secondary servers are dropped. If the coordinator fails, then one of the secondary servers must be elected as the new coordinator.

To enable this behavior in an application, the adapter or event bean, usually in the role of an event sink, must implement the following interface:

`com.bea.wlevs.ede.api.cluster.GroupMembershipListener`

This interface enables the event sink to listen for multiserver domain group membership changes. At runtime, Oracle Event Processing calls the `onMembershipChange` callback method whenever membership changes occur.

The callback method signature follows:

```
onMembershipChange(Server localIdentity, Configuration groupConfiguration);
```

In the implementation of the `onMembershipChange` callback method, the event sink uses the `Server` object (`localIdentity`) to verify if it is the leader. This can be done be comparing `localIdentity` with the result of `Configuration.getCoordinator()` run on the second parameter, `groupConfiguration`. This parameter also allows a server to know what the current members of the group are by executing `Configuration.getMembers()`.

To only keep events when it is a coordinator, the event sink must get a new `Server` identity every time membership in the group changes. Group membership changes occur when, for example, another server within the group fails and is no longer the coordinator.

The following interface is for listening to membership changes to the domain as a whole, rather than changes to the server group.

`com.bea.wlevs.ede.api.cluster.DomainMembershipListener`

In a hot-hot configuration, there is a non-zero delay in failure notification. If you use the notification APIs to implement clustering, you will lose and not process events that occur in the window of time between the server failure and the notification being delivered to the new master server.

See *Java API Reference for Oracle Event Processing*.

6.7 Start and Stop a Server in a Multiserver Domain

To start the servers in a multiserver domain, start each server separately by running its start script. This is the same way you start a server in a standalone server domain. See Start and Stop a Server in a Standalone-Server Domain for details.

If you have not configured custom groups for the multiserver domain, then all servers are members of just the predefined domain group, which contains all the servers in the multiserver domain, and a singleton group, one for each member server. This means, for example, if there are three servers in the multiserver domain then there are three singleton groups.

If, however, you have configured custom groups for the multiserver domain, then the servers are members of the groups for which they have been configured, as well as the pre-defined groups.

7

Multiserver Domain Application Deployment

You can deploy applications to Oracle Event Processing multiserver domains with either Oracle Event Processing Visualizer or the Deployer utility. This chapter describes how to use the Deployer utility.

See *Using Visualizer for Oracle Event Processing* for information about using Visualizer to deploy an application.

This chapter includes the following sections:

- Target Server Groups
- Deploy to a Server Singleton Group
- Deploy to a Server Domain Group
- Deploy to a Server Custom Group
- Troubleshooting.

7.1 Target Server Groups

When you deploy an application to a multiserver domain, you typically specify a target group, and Oracle Event Processing deploys the application to the set of running servers in that group. Oracle Event Processing dynamically maintains group membership based on running servers. When new servers in the group start, Oracle Event Processing propagates the appropriate set of deployments to the new server.

For example, with the multiserver domain configured in Multiserver Domains with Oracle Coherence, assume that only `myServer1` has been started. You then deploy an application to the domain group that includes `myServer1` and `myServer2`. Because only `myServer1` of the multiserver domain is running, the application deploys to `myServer1` only. When you start `myServer2`, Oracle Event Processing replicates and propagates the application deployment to `myServer2`.

Deployment propagation occurs based on the application version. When you deploy a new version of an application, the new version propagates to all servers in the group.

For more information, see:

- Server Groups
- Deployer Command-Line Reference
- *Using Visualizer for Oracle Event Processing*.

7.2 Deploy to a Server Singleton Group

If you do not specify a group when you deploy an application, Oracle Event Processing deploys the application to the singleton server group that includes only the specific server to which you deploy the application. This is the standard case in standalone-server domains, but is also applicable to multiserver domains.

The following example shows how to deploy to a singleton group. In this case, the command does not specify a `-group` option.

```
java -jar wlevsdeploy.jar -url http://ariel:9002/wlevsdeployer -install myapp_1.0.jar
```

The `myapp_1.0.jar` application deploys to the singleton server group running on host `ariel` and listening to port `9002`. If the domain is multiserver and other servers are members of the domain group, the application is not deployed to these servers.

7.3 Deploy to a Server Domain Group

The domain group is a live group that always exists and contains all servers in a domain. All servers are always a member of the domain group. However, you must still explicitly deploy applications to the domain group. The main reason for this is for simplicity and usage consistency. When you explicitly deploy an application to the domain group, Oracle Event Processing guarantees that all servers of this homogenous environment have this deployment.

To deploy to the domain group, use the `-group all` option. The following example shows how to deploy to a domain group.

```
java -jar wlevsdeploy.jar -url http://ariel:9002/wlevsdeployer -install
myapp_1.0.jar -group all
```

The `myapp_1.0.jar` application deploys to all servers in the domain group through the host `ariel` listen to port `9002`.

7.4 Deploy to a Server Custom Group

To deploy to a custom group, use the `-group` *`groupname`* option of the deploy command. In the following examples, assume the multiserver domain has been configured as described in Multiserver Domains with Oracle Coherence.

The following example shows how to deploy an application called `strategies_1.0.jar` to the `strategygroup`. Based on the multiserver domain configuration, the command deploys the application to `myServer2` and `to myServer3,which a`re the members of `strategygroup`.

```
java -jar wlevsdeploy.jar -url http://ariel:9002/wlevsdeployer -install
strategies_1.0.jar -group strategygroup
```

The following example shows how to deploy the `selector_1.0.jar` application to `selectorgroup`:

```
java -jar wlevsdeploy.jar -url http://ariel:9002/wlevsdeployer -install
selector_1.0.jar -group selectorgroup
```

Based on the multiserver domain configuration, the preceding command deploys the application to `myServer1` only, which is the sole member of `selectorgroup`.

Both commands deploy to the same server on host `ariel` listening to port `9002`. You can specify any of the servers in the domain in the deploy command, even if the server is not part of the group to which you want to deploy the application.

7.5 Troubleshooting

Oracle Event Processing server stops the application after deployment.

Problem: After you deploy an application to an Oracle Event Processing multiserver domain, Oracle Event Processing stops the application after about 30 seconds.

Solution: Be sure you do not have more than one VPN software package installed on the same computer hosting your multiserver domain.

Part IV

Configure Services

Configure Services contains the following chapters:

- Network I/O
- Security
- Jetty
- JMX
- JDBC
- HTTP Publish-Subscribe Server
- Logging and Debugging

8

Network I/O

Oracle Event Processing supports network I/O over TCP/IP with a variety of providers in server and client mode. You can define a network I/O service for SSL and non-SSL network access in the server `config.xml` file.

Oracle Event Processing servers are certified for use with IPv4 only or the IPv4/IPv6 dual-stack. For information about IPv6, see RFC 2460: Internet Protocol, Version 6 (IPv6) Specification at `http://www.ietf.org/rfc/rfc2460.txt`.

The `jetty` and `weblogic-rmi-client` server services depend on network I/O configuration. The `jetty` service depends on network I/O server (`netio`) configuration, and the `weblogic-rmi-client` service depends on network I/O client (`netio-client`) configuration.

This chapter includes the following sections:

8.1 Network I/O Providers

The following table lists the network I/O providers that Oracle Event Processing supports.

Table 8-1 Oracle Event Processing Network I/O Providers

provider-type	SSL?	Description
`non-blocking`	No	Provides fully non-blocking I/O for reads and writes. Each call to `read` or `write` on the `Connection` interface returns immediately without blocking. If the underlying connection is not ready, then the `read` or `write` call returns zero. At that point, the calling code must use one of the notification mechanisms in the NetIO API to wait until the connection is ready to read or write. Non-Blocking providers can also support a non-blocking `connect` call where a thread need not block if it takes a long time to establish (or if it fails to establish) a connection to a remote server.
`semi-blocking`	No	Provides non-blocking I/O for the `read` call, but each `write` call blocks until the data is handed to the TCP/IP stack. Some platforms enable you to implement a write-blocking provider that is faster than a fully non-blocking provider, but still allows for high scalability.

provider-type	SSL?	Description
`blocking`	No	Blocks on each `read` and `write` call until it completes. If there is no data to read, then `read` blocks until there is. This provider is much less scalable because there must a thread must wait for each network connection that might have data. Oracle recommends that you not use this type of provider.
`native`	No	Oracle Event Processing tries the `NativeAsyncEngine`, and if it is not supported, then raises an error.
`NIO`	Yes	The `NIOEngine` is always used. This is the default provider type.

The following example shows how to specify a provider in the `config.xml` file `netio` element using the `provider-type` child element.

```
<netio>
    <name>myNetio</name>
    <port>12345</port>
    <provider-type>non-blocking</provider-type>
</netio>
```

8.2 Configure Network I/O Server (netio)

You can define a network I/O service to be used by other services to act as the server and listen for incoming connections. You can also create a client network I/O service as Configure Network I/O Client (netio-client) describes.

You configure network I/O server services with the `netio` element in the Oracle Event Processing server `config.xml` file. For more information, see:

- Server Configuration Files
- Network I/O Providers
- *Schema Reference for Oracle Event Processing*.

Configure Network I/O Server

1. In the Oracle Event Processing server `config.xml` file, create a `netio` element:

   ```
   <netio>
   </netio>
   ```

2. Add a `name` element that uniquely identifies this `netio` element on this Oracle Event Processing server:

   ```
   <netio>
       <name>MyNetIO</name>
   </netio>
   ```

3. Add a `port` element to define the TCP/IP port on which this `netio` service listens for connection requests:

   ```
   <netio>
       <name>MyNetIO</name>
       <port>9002</port>
   </netio>
   ```

4. Optionally, specify a `provider-type`:

```
<netio>
    <name>MyNetIO</name>
    <port>9002</port>
    <provider-type>NIO</provider-type>
</netio>
```

5. Optionally, specify the other `netio` child elements.

   See *Schema Reference for Oracle Stream Explorer*.

8.3 Configure Network I/O Client (netio-client)

You can define a network I/O service to use to perform non-blocking network I/O, but that does not act as a server and does not listen for incoming connections. You can also create a server network I/O service as Configure Network I/O Server (netio) describes.

You configure network I/O client services with the `netio-client` element in the Oracle Event Processing server `config.xml` file.

For more information, see:

- Server Configuration Files
- Network I/O Providers
- *Schema Reference for Oracle Stream Explorer*.

To configure network I/O client:

1. In the Oracle Event Processing server `config.xml` file, create a `netio-client` element:

   ```
   <netio-client>
   </netio-client>
   ```

2. Add a `name` element that uniquely identifies this `netio` element on this Oracle Event Processing server:

   ```
   <netio-client>
       <name>MyNetIOClient</name>
   </netio-client>
   ```

3. Optionally, specify a `provider-type`:

   ```
   <netio-client>
       <name>MyNetIOClient</name>
       <provider-type>NIO</provider-type>
   </netio-client>
   ```

4. Optionally, specify the other `netio-client` child elements.

   See *Schema Reference for Oracle Stream Explorer*.

9

Security

Oracle Event Processing provides a variety of ways to protect server resources such as data and event streams, configuration, user name and password data, security policy information, remote credentials, and network traffic.

This chapter contains the following sections:

9.1 Users, Groups, and Roles

Oracle Event Processing uses role-based authorization control to secure the Oracle Event Processing Visualizer and the `wlevs.Admin` command-line utility. There are a variety of default out-of-the-box security groups. You can add users to different groups to give them the different roles.

Administrators who use Oracle Event Processing Visualizer, `wlevs.Admin`, or any custom administration application that uses JMX to connect to an Oracle Event Processing server use role-based authorization to gain access.

You can also use role-based authorization to control access to the HTTP publish-subscribe server.

There are two types of roles:

- Application roles: Application roles grant users the permission to access various Oracle CQL applications deployed to the Oracle Event Processing server. You can create application roles and associate them with the task roles that Oracle Event Processing provides.

  By default, administrator users can access any application and non-administration users cannot access any applications. Before a non-administration user can access an application, an administration user must grant the user the associated application role.

  Application Isolation enables administrators to create new roles that can associate new roles to a given application to allow only a selected group access to this application. After the new role is created, non-admin users without this role cannot see the application in visualizer and cannot list or change the application configuration through Admin tool.

- Task roles: Task roles grant users the permission to perform various tasks with the applications their application role authorizes them to access. Oracle Event Processing provides the default task roles that Table 9-1 describes.

Users that successfully authenticate themselves when using Oracle Event Processing Visualizer or `wlevs.Admin` are assigned roles based on their group membership, and then subsequent access to administrative functions is restricted according to the roles held by the user. Anonymous users (non-authenticated users) will not have any access to the Oracle Event Processing Visualizer or `wlevs.Admin`.

When an administrator uses the Configuration Wizard to create a new domain, he or she enters an administrator user that is part of the `wlevsAdministrators` group. By default, this information is stored in a file-based provider filestore. The password is hashed using the SHA-256 algorithm. The default administrator user is named `oepadmin` with password `welcome1`.

Table 9-1 describes the default Oracle Event Processing tasks roles available right after the creation of a new domain, as well as the name of the groups that are assigned to these roles.

Table 9-1 Default Oracle Event Processing Task Roles and Groups

Task Role	Group	Privileges
`Admin`	`wlevsAdministrators`	Has all privileges of all the other roles, and permission to: • Create users and groups • Configure HTTP publish-subscribe security • Change the system configuration, such as Jetty, work manager, and so on. All JMX `get`, `set`, `start`, `stop`, and `deploy` operations. All `set` and `invoke` methods on these MBeans: `CQLProcessorMBeanCQLProcessorRuntimeMBeanRecordPlaybackMBeanChannelMBeanProfileMBeanProfileManagerBean`

Task Role	Group	Privileges
`ApplicationAdmin`	`wlevsApplicationAdmins`	Has all Operator privileges and permission to update the configuration of any deployed application. All `set` and `invoke` methods on `ChannelMBean.`
`BusinessUser`	`wlevsBusinessUsers`	Has all Operator privileges and permission to update the Oracle CQL rules associated with the processor of a deployed application. All `set` and `invoke` methods on these MBeans: `CQLProcessorMBeanCQLProcessorRuntimeMBean`
`Deployer`	`wlevsDeployers`	Has all Operator privileges and permission to deploy, undeploy, update, suspend, and resume any deployed application. All JMX `get` and `set` operations related to deployment.
`Monitor`	`wlevsMonitors`	Has all Operator privileges and permission to enable/disable diagnostic functions, such as creating a diagnostic profile and recording events (then playing them back.) Can also inject and trace events. All JMX `get` operations. All `set` and `invoke` methods on these MBeans: `RecordPlaybackMBeanProfileMBeanProfileManagerBean`
`Operator`	`wlevsOperators`	Has read-only access to all server resources, services, and deployed applications.

Once the domain is created, the administrator can use Oracle Event Processing Visualizer to create a group and associate it with one or more roles. Each role grants access to an application. When you assign a user to a group, the roles you associate with the group give the user the privileges to access those applications.

For instructions on using Oracle Event Processing Visualizer to modify users, groups, and roles, see *Using Visualizer for Oracle Stream Explorer*

For more information, see:

- Configure HTTP Publish-Subscribe Server Channel Security
- User Credentials for Command-Line Utilities.

9.2 Java SE Security for an Oracle Event Processing Server

The Java SE platform defines a standards-based and interoperable security architecture that is dynamic and extensible. Security for features such as cryptography, authentication and authorization, public key encryption, and more are built in. See `http://java.sun.com/javase/technologies/security/` for more information.

Oracle Event Processing supports Java SE security with the following security policy files. Oracle provides templates for these files in the product in the following directory: */Oracle/Middleware/my_oep/*oep/utils/security.

- `policy.xml`: Defines the security policies of all the bundles that make up Oracle Event Processing. The first bundle set defines the policies for server-related bundles; the second bundle set defines the policies for application bundles.
- `security.policy`: Defines the security policies for server startup and Web applications deployed to the Jetty HTTP server. This file also defines policies for the Oracle Event Processing Visualizer Web application.

You can define Java SE security policies for the following Oracle Event Processing features:

- All bundles that make up Oracle Event Processing
- Server startup
- Web applications deployed to the Oracle Event Processing server Jetty HTTP server
- Oracle Event Processing Visualizer

Configure Java SE Security on a Server:

1. Stop the Oracle Event Processing server, if it is currently running.

   See Start and Stop Servers.

2. Copy `policy.xml` and `security.policy`:

   From:*/Oracle/Middleware/my_oep/oep/utils/security/*

   To:

   /Oracle/Middleware/my_oep/user_projects/domains/
   <domainname>/<*servername*>/config/

3. Edit the two security policy files as needed.

4. Update the server startup script for your platform in the <*servername*> directory by adding the following properties to the `java` command that starts the server:

   ```
   -Djava.security.manager
   -Djava.security.policy=./config/security.policy
   -Dcom.bea.core.security.policy=./config/policy.xml
   ```

   For example with everything on one line:

   ```
   "%JAVA_HOME%\bin\java" %DGC% %DEBUG% -Djava.security.manager
   -Djava.security.policy=./config/security.policy
   -Dcom.bea.core.security.policy=./config/policy.xml
   -Dwlevs.home="%USER_INSTALL_DIR%" -Dbea.hoe="%BEA_HOME%"
   -jar "%USER_INSTALL_DIR%\bin\wlevs.jar" %1 %2 %3 %4 %5 %6
   ```

5. Edit the Jetty configuration in the `config.xml` server file by adding a `<scratch-directory>` child element of the `<jetty>` element to specify the directory to which Jetty Web applications are deployed.

   For example:

```
<jetty>
    <name>JettyServer</name>
    <network-io-name>NetIO</network-io-name>
    <work-manager-name>JettyWorkManager</work-manager-name>
    <secure-network-io-name>sslNetIo</secure-network-io-name>
    <scratch-directory>./JettyWork</scratch-directory>
</jetty>
```

6. Restart the Oracle Event Processing server for the changes to take effect.

   See Start and Stop Servers.

9.3 Security Provider

Oracle Event Processing supports the following security providers for authentication, authorization, role mapping, and credential mapping. The default is the file-based provider. If you use the default file-based security provider, then you do not need to do any further configuration of your domain. If you want to use the LDAP or DBMS providers, you must perform further configuration. Once you have configured the security provider, you can add new users, assign them to groups, and map groups to roles. See Users, Groups, and Roles.

- File-based: Default security provider that uses an operating system file to access security data such as user, password, and group information. This provider provides authentication and authorization. Authentication is the process whereby the identity of users is proved or verified. Authorization is the process whereby a user's access to an Oracle Event Processing resource is permitted or denied based on the user's security role and the security policy assigned to the requested Oracle Event Processing resource. Authentication typically involves user name and password combinations.
- LDAP: Uses a Lightweight Data Access Protocol (LDAP) server to access user, password, and group information. Provides only authentication.

  When you use LDAP for authentication, you cannot add or delete users and groups using Oracle Event Processing Visualizer, you can only change the password of a user.
- RDBMS: Uses a DBMS to access user, password, and group information. Provides both authentication and authorization.

The following procedures describe two different ways to configure a security provider for authentication and authorization.

- Configure Authentication with LDAP and Authorization with the DBMS Provider
- Configure Authentication and Authorization with the DBMS Provider

Configure Authentication with LDAP and Authorization with the DBMS Provider

1. Add the *Oracle/Middleware/my_oep/*`oep/bin` directory to your `PATH` environment variable:

   ```
   set PATH=d:\Oracle\Middleware\my_oep\oep\bin;%PATH% (Windows)
   PATH=/Oracle/Middleware/my_oep/oep/bin:$PATH (UNIX)
   ```

2. Go to the `config` directory for the server you want to configure.

   By default, the `config` directory is in */Oracle/Middleware/my_oep/ user-Projects/domains/<domainname>/<servername>/*`config/`.

3. In a text editor, create a file called `myLDAPandDBMS.properties` and copy the entire contents of the following example into it.

Note:

Make sure the certificate of the boot user in the `evsidentity.jks` file is the same as what is configured in the `security.xml` file.

```
# For attributes of type boolean or Boolean, value can be "true" or "false"
# and it's case insensitive.
# For attributes of type String[], values are comma separated; blanks before
# and after the comma are ignored. For example, if the property is defined as:
#   saml1.IntersiteTransferURIs=uri1, uri2, uri3
# the IntersiteTransferURIs attribute value is String[]{"uri1", "uri2", "uri3"}
# For attributes of type Properties, the value should be inputted as
# a set of key=value pairs separated by commas; blanks before and after the
# commas are also ignored. For example (in practice, the property should be all on one
line):
#  store.StoreProperties=DriverName=oracle.jdbc.driver.OracleDriver,
ConnectionURL=jdbc:oracle:thin:@united.bea.com:1521:xe, Username=user, Password=user
domain.mbean=com.bea.common.management.configuration.LegacyDomainInfoMBean
domain.DomainName=legacy-domain-name
domain.ServerName=legacy-server-name
domain.RootDirectory=legacy-rootdir
#domain.ProductionModeEnabled=
#domain.WebAppFilesCaseInsensitive=
domain.DomainCredential=changeit
jaxp.mbean=com.bea.common.management.configuration.JAXPFactoryServiceMBean
#jaxp.DocBuilderFactory=
#jaxp.SaxParserFactory=
#jaxp.SaxTransformFactory=
#jaxp.TransformFactory=
#ldapssl.mbean=com.bea.common.management.configuration.LDAPSSLSocketFactoryLookupServiceMBe
an
#ldapssl.Protocol=sslv3
#ldapssl.TrustManagerClassName=
namedsql.mbean=com.bea.common.management.configuration.NamedSQLConnectionLookupServiceMBean
store.mbean=com.bea.common.management.configuration.StoreServiceMBean
# Split here for readability; in practice, a property should be all on one line.
store.StoreProperties=DriverName=oracle.jdbc.driver.OracleDriver,
    ConnectionURL=jdbc:oracle:thin:@localhost:1521:orcl, Username=wlevs, Password=wlevs
#store.ConnectionProperties=
#store.NotificationProperties=
realm.mbean=weblogic.management.security.RealmMBean
realm.Name=my-realm
#realm.ValidateDDSecurityData=
#realm.CombinedRoleMappingEnabled=
#realm.EnableWebLogicPrincipalValidatorCache=
#realm.MaxWebLogicPrincipalsInCache=
#realm.DelegateMBeanAuthorization=
#realm.AuthMethods=
adt.1.mbean=weblogic.security.providers.audit.DefaultAuditorMBean
adt.1.Severity=INFORMATION
#adt.1.InformationAuditSeverityEnabled=
#adt.1.WarningAuditSeverityEnabled=
#adt.1.ErrorAuditSeverityEnabled=
#adt.1.SuccessAuditSeverityEnabled=
#adt.1.FailureAuditSeverityEnabled=
#adt.1.OutputMedium=
#adt.1.RotationMinutes=
#adt.1.BeginMarker=
#adt.1.EndMarker=
#adt.1.FieldPrefix=
#adt.1.FieldSuffix=
adt.1.Name=my-auditor
```

```
#adt.1.ActiveContextHandlerEntries=
atn.1.mbean=weblogic.security.providers.authentication.LDAPAuthenticatorMBean
#atn.1.UserObjectClass=
#atn.1.UserNameAttribute=
#atn.1.UserDynamicGroupDNAttribute=
atn.1.UserBaseDN=o=ECS,dc=bea,dc=com
atn.1.UserSearchScope=subtree
#atn.1.UserFromNameFilter=
#atn.1.AllUsersFilter=
atn.1.GroupBaseDN=ECS,dc=bea,dc=com
#atn.1.GroupSearchScope=
#atn.1.GroupFromNameFilter=
#atn.1.AllGroupsFilter=
#atn.1.StaticGroupObjectClass=
#atn.1.StaticGroupNameAttribute=
atn.1.StaticMemberDNAttribute=member
#atn.1.StaticGroupDNsfromMemberDNFilter=
#atn.1.DynamicGroupObjectClass=
#atn.1.DynamicGroupNameAttribute=
#atn.1.DynamicMemberURLAttribute=
atn.1.GroupMembershipSearching=unlimited
atn.1.MaxGroupMembershipSearchLevel=0
atn.1.UseRetrievedUserNameAsPrincipal=false
#atn.1.IgnoreDuplicateMembership=
#atn.1.KeepAliveEnabled=
atn.1.Credential=wlevs
#atn.1.Name=
#atn.1.PropagateCauseForLoginException=
atn.1.ControlFlag=REQUIRED
#atn.1.ConnectTimeout=
atn.1.Host=localhost
atn.1.Port=389
#atn.1.SSLEnabled=
atn.1.Principal=cn=Administrator,dc=bea,dc=com
#atn.1.CacheEnabled=
#atn.1.CacheSize=
#atn.1.CacheTTL=
atn.1.FollowReferrals=false
#atn.1.BindAnonymouslyOnReferrals=
#atn.1.ResultsTimeLimit=
#atn.1.ParallelConnectDelay=
#atn.1.ConnectionRetryLimit=
atn.1.EnableGroupMembershipLookupHierarchyCaching=true
#atn.1.MaxGroupHierarchiesInCache=
#atn.1.GroupHierarchyCacheTTL=
#atn.5.mbean=weblogic.security.providers.authentication.OpenLDAPAuthenticatorMBean
#atn.5.UserNameAttribute=
#atn.5.UserBaseDN=
#atn.5.UserFromNameFilter=
#atn.5.GroupBaseDN=
#atn.5.GroupFromNameFilter=
#atn.5.StaticGroupObjectClass=
#atn.5.StaticMemberDNAttribute=
#atn.5.StaticGroupDNsfromMemberDNFilter=
#atn.5.UserObjectClass=
#atn.5.UserDynamicGroupDNAttribute=
#atn.5.UserSearchScope=
#atn.5.AllUsersFilter=
#atn.5.GroupSearchScope=
#atn.5.AllGroupsFilter=
#atn.5.StaticGroupNameAttribute=
#atn.5.DynamicGroupObjectClass=
#atn.5.DynamicGroupNameAttribute=
#atn.5.DynamicMemberURLAttribute=
#atn.5.GroupMembershipSearching=
#atn.5.MaxGroupMembershipSearchLevel=
#atn.5.UseRetrievedUserNameAsPrincipal=
#atn.5.IgnoreDuplicateMembership=
#atn.5.KeepAliveEnabled=
```

```
#atn.5.Credential=
#atn.5.PropagateCauseForLoginException=
#atn.5.ControlFlag=
#atn.5.Name=
#atn.5.ConnectTimeout=
#atn.5.Host=
#atn.5.Port=
#atn.5.SSLEnabled=
#atn.5.Principal=
#atn.5.CacheEnabled=
#atn.5.CacheSize=
#atn.5.CacheTTL=
#atn.5.FollowReferrals=
#atn.5.BindAnonymouslyOnReferrals=
#atn.5.ResultsTimeLimit=
#atn.5.ParallelConnectDelay=
#atn.5.ConnectionRetryLimit=
#atn.5.EnableGroupMembershipLookupHierarchyCaching=
#atn.5.MaxGroupHierarchiesInCache=
#atn.5.GroupHierarchyCacheTTL=
cm.1.mbean=weblogic.security.providers.credentials.DefaultCredentialMapperMBean
cm.1.Name=my-credential-mapper
cm.1.CredentialMappingDeploymentEnabled=true
#cm.3.mbean=weblogic.security.providers.credentials.FileBasedCredentialMapperMBean
#cm.3.FileStorePath=
#cm.3.FileStorePassword=
#cm.3.EncryptAlgorithm=
#cm.3.Name=
#cm.3.CredentialMappingDeploymentEnabled=
rm.1.mbean=weblogic.security.providers.xacml.authorization.XACMLRoleMapperMBean
rm.1.Name=my-role-mapper
rm.1.RoleDeploymentEnabled=true
atz.1.mbean=weblogic.security.providers.xacml.authorization.XACMLAuthorizerMBean
atz.1.Name=my-authorizer
atz.1.PolicyDeploymentEnabled=true
adj.1.mbean=weblogic.security.providers.authorization.DefaultAdjudicatorMBean
adj.1.RequireUnanimousPermit=false
adj.1.Name=my-adjudicator
```

   a. Customize the property file by updating the `store.StoreProperties` property to reflect your database driver information, connection URL, and user name and password of the user that connects to the database. This is how the default property is set:

      ```
      # Split for readability; in practice, the property should be on one line.
      store.StoreProperties=DriverName=oracle.jdbc.driver.OracleDriver,
      ConnectionURL=jdbc:oracle:thin:@mymachine:1521:orcl, Username=wlevs,
      Password=wlevs
      ```

   b. Update the property that specifies your LDAP server configuration.

   c. Leave all the other properties to their default values.

4. Make a backup copy of the existing `security.xml` file in case you need to revert.

5. Create a new security configuration file (`security.xml`) by executing the following `cssconfig` command:

   ```
   cssconfig -p myLDAPandDBMS.properties -c security.xml -i security-key.dat
   ```

   `myLDAPandDBMS.properties`: The property file you created in step 3.
   `security.xml`: The name of the new security configuration file.`security-key.dat`: An existing file generated by the Configuration Wizard that contains the identity key.

See The cssconfig Command-Line Utility for additional information.

6. Go to */Oracle/Middleware/my_oep/*oep/utils/security/sql.

   This directory contains SQL scripts for creating the required security-related database tables and populating them with initial data. Because you are using the DBMS provider only for authorization, the relevant scripts for this procedure are:

   atz_create.sql: Creates all tables required for authorization.

   atz_drop.sql: Drops all authorization-related tables.

7. Run the atz_create.sql SQL script against the database you specified as the database store in step 3.

8. Configure your LDAP server by adding the default groups described in Users, Groups, and Roles and the administrator user you specified when you created the domain. By default, this user is called oepadmin.

   Refer to your LDAP server documentation for details.

9. Optionally, configure password strength in your new security.xml file.

   See Password Strength.

Configure Authentication and Authorization with the DBMS Provider

1. Add the */Oracle/Middleware/my_oep/*oep/bin directory to your PATH environment variable:

```
prompt> set PATH=d:\Oracle\Middleware\my_oep\oep\bin;%PATH% (Windows)
prompt> PATH=/Oracle/Middleware/my_oep/oep/bin:$PATH (UNIX)
```

2. Go to the config directory for the server you want to configure.

   By default the config directory is in */Oracle/Middleware/my_oep/ user-Projects/domains/<domainname>/<servername>/*config/.

3. Make a backup copy of the existing security.xml file, in case you need to revert.

4. In a text editor, create a file called myDBMS.properties and copy the entire contents of the following example into it.

```
# For attributes of type boolean or Boolean, value can be "true" or "false"
# and it's case insensitive.
# For attributes of type String[], values are comma separated; blanks before
# and after the comma are ignored. For example, if the property is defined as:
#   saml1.IntersiteTransferURIs=uri1, uri2, uri3
# the IntersiteTransferURIs attribute value is String[]{"uri1", "uri2", "uri3"}
# For attributes of type Properties, the value should be inputted as
# a set of key=value pairs separated by commas; blanks before and after the
# commas are also ignored. For example (split for readability; in practice, the property
should be all on one line):
#  store.StoreProperties=DriverName=oracle.jdbc.driver.OracleDriver,
    ConnectionURL=jdbc:oracle:thin:@united.bea.com:1521:xe, Username=user, Password=user
domain.mbean=com.bea.common.management.configuration.LegacyDomainInfoMBean
domain.DomainName=legacy-domain-name
domain.ServerName=legacy-server-name
domain.RootDirectory=legacy-rootdir
#domain.ProductionModeEnabled=
#domain.WebAppFilesCaseInsensitive=
domain.DomainCredential=changeit
jaxp.mbean=com.bea.common.management.configuration.JAXPFactoryServiceMBean
#jaxp.DocBuilderFactory=
#jaxp.SaxParserFactory=
```

```
#jaxp.SaxTransformFactory=
#jaxp.TransformFactory=
#ldapssl.mbean=com.bea.common.management.configuration.LDAPSSLSocketFactoryLookupServiceMBe
an
#ldapssl.Protocol=
#ldapssl.TrustManagerClassName=
namedsql.mbean=com.bea.common.management.configuration.NamedSQLConnectionLookupServiceMBean
store.mbean=com.bea.common.management.configuration.StoreServiceMBean
# Split for readability; the property should be fully on one line.
store.StoreProperties=DriverName=oracle.jdbc.driver.OracleDriver,
    ConnectionURL=jdbc:oracle:thin:@mymachine:1521:orcl, Username=wlevs, Password=wlevs
#store.ConnectionProperties=
#store.NotificationProperties=
realm.mbean=weblogic.management.security.RealmMBean
realm.Name=my-realm
#realm.ValidateDDSecurityData=
#realm.CombinedRoleMappingEnabled=
#realm.EnableWebLogicPrincipalValidatorCache=
#realm.MaxWebLogicPrincipalsInCache=
#realm.DelegateMBeanAuthorization=
#realm.AuthMethods=
sqlconn.1.mbean=com.bea.common.management.configuration.NamedSQLConnectionMBean
sqlconn.1.Name=POOL1
sqlconn.1.JDBCDriverClassName=oracle.jdbc.driver.OracleDriver
sqlconn.1.ConnectionPoolCapacity=5
sqlconn.1.ConnectionPoolTimeout=10000
sqlconn.1.AutomaticFailoverEnabled=false
sqlconn.1.PrimaryRetryInterval=0
sqlconn.1.JDBCConnectionURL=jdbc\:oracle\:thin\:@fwang02\:1521\:orcl
sqlconn.1.JDBCConnectionProperties=
sqlconn.1.DatabaseUserLogin=wlevs
sqlconn.1.DatabaseUserPassword=wlevs
sqlconn.1.BackupJDBCConnectionURL=
sqlconn.1.BackupJDBCConnectionProperties=
sqlconn.1.BackupDatabaseUserLogin=
sqlconn.1.BackupDatabaseUserPassword=
adt.1.mbean=weblogic.security.providers.audit.DefaultAuditorMBean
adt.1.Severity=INFORMATION
#adt.1.InformationAuditSeverityEnabled=
#adt.1.WarningAuditSeverityEnabled=
#adt.1.ErrorAuditSeverityEnabled=
#adt.1.SuccessAuditSeverityEnabled=
#adt.1.FailureAuditSeverityEnabled=
#adt.1.OutputMedium=
#adt.1.RotationMinutes=
#adt.1.BeginMarker=
#adt.1.EndMarker=
#adt.1.FieldPrefix=
#adt.1.FieldSuffix=
adt.1.Name=my-auditor
#adt.1.ActiveContextHandlerEntries=
atn.1.mbean=weblogic.security.providers.authentication.SQLAuthenticatorMBean
atn.1.PasswordAlgorithm=SHA-1
atn.1.PasswordStyle=SALTEDHASHED
atn.1.PasswordStyleRetained=true
atn.1.SQLCreateUser=INSERT INTO USERS VALUES ( ? , ? , ? )
atn.1.SQLRemoveUser=DELETE FROM USERS WHERE U_NAME \= ?
atn.1.SQLRemoveGroupMemberships=DELETE FROM GROUPMEMBERS WHERE G_MEMBER \= ? ORG_NAME \= ?
atn.1.SQLSetUserDescription=UPDATE USERS SET U_DESCRIPTION  \= ? WHERE U_NAME \= ?
atn.1.SQLSetUserPassword=UPDATE USERS SET U_PASSWORD \= ? WHERE U_NAME \= ?
atn.1.SQLCreateGroup=INSERT INTO GROUPS VALUES ( ? , ? )
atn.1.SQLSetGroupDescription=UPDATE GROUPS SET G_DESCRIPTION \= ? WHERE G_NAME \=  ?
atn.1.SQLAddMemberToGroup=INSERT INTO GROUPMEMBERS VALUES( ?, ?)
atn.1.SQLRemoveMemberFromGroup=DELETE FROM GROUPMEMBERS WHERE G_NAME \= ? AND G_MEMBER \= ?
atn.1.SQLRemoveGroup=DELETE FROM GROUPS WHERE G_NAME \= ?
atn.1.SQLRemoveGroupMember=DELETE FROM GROUPMEMBERS WHERE G_NAME \= ?
atn.1.SQLListGroupMembers=SELECT G_MEMBER FROM GROUPMEMBERS WHERE G_NAME \= ? AND G_MEMBER
LIKE ?
atn.1.DescriptionsSupported=true
```

```
atn.1.SQLGetUsersPassword=SELECT U_PASSWORD FROM USERS WHERE U_NAME \= ?
atn.1.SQLUserExists=SELECT U_NAME FROM USERS WHERE U_NAME \= ?
atn.1.SQLListMemberGroups=SELECT G_NAME FROM GROUPMEMBERS WHERE G_MEMBER \= ?
atn.1.SQLListUsers=SELECT U_NAME FROM USERS WHERE U_NAME LIKE ?
atn.1.SQLGetUserDescription=SELECT U_DESCRIPTION FROM USERS WHERE U_NAME \= ?
atn.1.SQLListGroups=SELECT G_NAME FROM GROUPS WHERE G_NAME LIKE ?
atn.1.SQLGroupExists=SELECT G_NAME FROM GROUPS WHERE G_NAME \= ?
atn.1.SQLIsMember=SELECT G_MEMBER FROM GROUPMEMBERS WHERE G_NAME \= ? AND G_MEMBER \= ?
atn.1.SQLGetGroupDescription=SELECT G_DESCRIPTION FROM GROUPS WHERE G_NAME \= ?
atn.1.GroupMembershipSearching=unlimited
atn.1.MaxGroupMembershipSearchLevel=0
atn.1.DataSourceName=POOL1
atn.1.PlaintextPasswordsEnabled=true
atn.1.ControlFlag=REQUIRED
atn.1.Name=my-authenticator
atn.1.EnableGroupMembershipLookupHierarchyCaching=false
atn.1.MaxGroupHierarchiesInCache=100
atn.1.GroupHierarchyCacheTTL=60
cm.1.mbean=weblogic.security.providers.credentials.DefaultCredentialMapperMBean
cm.1.Name=my-credential-mapper
cm.1.CredentialMappingDeploymentEnabled=true
rm.1.mbean=weblogic.security.providers.xacml.authorization.XACMLRoleMapperMBean
rm.1.Name=my-role-mapper
rm.1.RoleDeploymentEnabled=true
atz.1.mbean=weblogic.security.providers.xacml.authorization.XACMLAuthorizerMBean
atz.1.Name=my-authorizer
atz.1.PolicyDeploymentEnabled=true
adj.1.mbean=weblogic.security.providers.authorization.DefaultAdjudicatorMBean
adj.1.RequireUnanimousPermit=false
adj.1.Name=my-adjudicator
```

a. Customize the property file by updating the `store.StoreProperties` property to reflect your database driver information, connection URL, and user name and password of the user that connects to the database. This is how the default property is set:

```
store.StoreProperties=DriverName=oracle.jdbc.driver.OracleDriver,
ConnectionURL=jdbc:oracle:thin:@mymachine:1521:orcl, Username=wlevs,
Password=wlevs
```

b. Leave all the other properties to their default values.

5. Create a new security configuration file (`security.xml`) by executing the following `cssconfig` command:

```
cssconfig -p myLDAPandDBMS.properties -c security.xml -i security-key.dat
```

`myDBMS.properties`: The property file you created in step 3.`security.xml`: The name of the new security configuration file. `security-key.dat`: An existing file generated by the Configuration Wizard that contains the identity key.

See The cssconfig Command-Line Utility for additional information.

6. Go to */Oracle/Middleware/my_oep/*`oep/utils/security/sql`:

This directory contains SQL scripts for creating the required security-related database tables and populating them with initial data. These scripts are:

`atn_create.sql`: Creates all tables required for authentication.

`atn_drop.sql`: Drops all authentication-related tables.

`atn_init.sql`: Inserts default values into the authentication-related user and group tables. In particular, the script inserts a single default administrator user called `oepadmin`, with password `welcome1`, into the user table and specifies that

the user belongs to the wlevsAdministrators group. The script also inserts the default groups listed in Table 9-1 into the group table.

`atz_create.sql`: Creates all tables required for authorization.

`atz_drop.sql`: Drops all authorization-related tables.

7. If, when you created your domain using the Configuration Wizard, you specified an administrator user other than the default, edit the `atn_init.sql` file and add the `INSERT INTO USERS` and corresponding `INSERT INTO GROUPMEMBERS` statements.

   For example, to add an administrative user `juliet`, with password `shackell`, add the following statements to the `atn_init.sql` file:

   ```
   INSERT INTO USERS (U_NAME, U_PASSWORD, U_DESCRIPTION) VALUES
   ('juliet','shackell','default admin');
   INSERT INTO GROUPMEMBERS (G_NAME, G_MEMBER) VALUES
   ('wlevsAdministrators','juliet');
   ```

8. Run the following SQL script files, in the order listed against the database you specified as the database store in step 3:

   `atn_create.sql`

   `atn_init.sql`

   `atz_create.sql`

9. Optionally, configure password strength in your new `security.xml` file.

   See Password Strength.

9.4 Password Strength

Password strength measures the effectiveness of a password as an authentication credential. How you configure password strength determines the type of password a user can specify, such as whether the password can contain the user name, the minimum length of the password, the minimum number of numeric characters it can contain, and so on.

You configure the strength of the passwords used for Oracle Event Processing authentication by updating the <password-validator> element in the security configuration file (`security.xml`).

By default, the security configuration file is in the *`Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config`* directory.

The following example shows a snippet from the `security.xml` file with the default values after you create a new domain.

```
<sec:password-validator
   xmlns:pas="http://www.bea.com/ns/weblogic/90/security/providers/passwordvalidator"
   xsi:type="pas:system-password-validatorType">
   <sec:name>my-password-validator</sec:name>
   <pas:reject-equal-or-contain-username>true</pas:reject-equal-or-contain-username>
   <pas:reject-equal-or-contain-reverse-username>
      false
   </pas:reject-equal-or-contain-reverse-username>
   <pas:max-password-length>50</pas:max-password-length>
   <pas:min-password-length>6</pas:min-password-length>
   <pas:max-instances-of-any-character>0</pas:max-instances-of-any-character>
   <pas:max-consecutive-characters>0</pas:max-consecutive-characters>
   <pas:min-alphabetic-characters>1</pas:min-alphabetic-characters>
   <pas:min-numeric-characters>1</pas:min-numeric-characters>
```

```
    <pas:min-lowercase-characters>1</pas:min-lowercase-characters>
    <pas:min-uppercase-characters>1</pas:min-uppercase-characters>
    <pas:min-non-alphanumeric-characters>0</pas:min-non-alphanumeric-characters>
</sec:password-validator>
```

Table 9-2 describes all the child elements of `<password-validator>` that you can configure. If you manually update the `security.xml` file, you must restart the Oracle Event Processing server instance for the changes to take effect.

Table 9-2 Child Elements of <password-validator>

Child Element	Description	Default Value
`reject-equal-or-contain-name`	When set to `true`, Oracle Event Processing rejects a password if it is the same as, or contains, the user name. When set to `false`, Oracle Event Processing does not reject a password for this reason.	`true`
`reject-equal-or-contain-reverse-username`	When set to `true`, Oracle Event Processing rejects a password if it is the same as, or contains, the reversed user name. When set to `false`, Oracle Event Processing does not reject a password for this reason.	`false`
`max-password-length`	Specifies the maximum length of a password. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	50
`min-password-length`	Specifies the minimum length of a password. Valid values for this element are integers greater than or equal to 0.	6
`max-instances-of-any-character`	Specifies the maximum number of times the same character can appear in the password. For example, if this element is set to 2, then the password `bubble` is invalid. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	0
`max-consecutive-characters`	Specifies the maximum number of repeating consecutive characters that are allowed in the password. For example, if this element is set to 2, then the password `bubbble` is invalid. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	0
`min-alphabetic-characters`	Specifies the minimum number of alphabetic characters that a password must contain. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	1

Child Element	Description	Default Value
`min-numeric-characters`	Specifies the minimum number of numeric characters that a password must contain. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	1
`min-lowercase-characters`	Specifies the minimum number of lowercase characters that a password must contain. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	0
`min-uppercase-characters`	Specifies the minimum number of uppercase characters that a password must contain. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	0
`min-non-alphanumeric-characters`	Specifies the minimum number of non-alphanumeric characters that a password must contain. Non-alphanumeric characters include `$`, `#`, `@`, `&`, `!` and so on. A value of 0 means there is no restriction. Valid values for this element are integers greater than or equal to 0.	0

9.5 SSL to Secure Network Traffic

Oracle Event Processing provides one-way Secure Sockets Layer (SSL) to secure network traffic as between:

- A browser running the Oracle Event Processing Visualizer and the Oracle Event Processing server that hosts the data-services application that the Oracle Event Processing Visualizer uses.
- The `wlevs.Admin` command-line utility and an Oracle Event Processing instance.

  See Run wlevs.Admin Utility in SSL Mode.
- The member servers of a multiserver domain.

You can configure Oracle Event Processing to use a Federal Information Processing Standards (FIPS)-certified pseudo-random number generator for SSL. After you configure SSL, you can configure the Oracle Event Processing server to accept only client requests on the HTTPS port. See HTTPS-Only Connections.

You configure SSL in the server's `config.xml` file. When you create an Oracle Event Processing server, the server `config.xml` includes a default SSL configuration. The following procedures show how to configure SSL and a key store.

- Configure SSL Manually
- Create a Key Store Manually
- Configure SSL Between an SAML2 Service Provider and Identity Provider
- Configure SSL Between an SAML2 Service Provider and Identity Provider.

9.5.1 Configure SSL Manually

1. Use the Configuration Wizard to create a standalone-server domain or a multiserver domain.

   See:

   Standalone-Server Domains

   Multiserver Domains with Oracle Coherence

   Multiserver Domains with Native Clustering

2. In an XML editor, open the `config.xml` file for the server you want to configure.

   By default, the `config.xml` file is in */Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config*.

3. Configure the `ssl` element.

   The following example shows the default `ssl` element the Configuration Wizard creates.

```
<ssl>
    <name>sslConfig</name>
    <key-store>./ssl/evsidentity.jks</key-store>
    <key-store-pass>
        <password>{Salted-3DES}sdlUX8aEDeNpQ4VhsaCnFA==</password>
    </key-store-pass>
    <key-store-alias>evsidentity</key-store-alias>
    <key-manager-algorithm>SunX509</key-manager-algorithm>
    <ssl-protocol>TLS</ssl-protocol>
    <enforce-fips>false</enforce-fips>
    <need-client-auth>false</need-client-auth>
</ssl>
```

   The `key-store` element points to a certificate file. The Configuration Wizard creates a default certificate file called `evsidentity.jks` in the `ssl` directory for the server you want to configure.

   By default, the password for the certificate private key is the same as the password for the identity key store.

   Note:

   The Oracle Event Processing Server will not start unless the password for the certificate private key is the same as the password for the identity key store.

   The `evsidentity.jks` file contains a self-signed certificate. Optionally, create your own certificate file and either replace the `evsidentity.jks` file, or update the `key-store` element in the `config.xml` file.

   Note:

   In a production environment, the system administrator should replace the default self-signed certificate with a CA signed certificate.

For more information about creating a key store, see Create a Key Store Manually.

For more information about the `enforce-fips` element, see FIPS.

4. Configure a `netio` element for SSL.

   The following example shows the default `netio` element the Configuration Wizard creates.

```
<netio>
    <name>sslNetIo</name>
    <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
    <port>9003</port>
</netio>
```

   The `ssl-config-bean-name` must match the `ssl` element `name` child element.

   The default secure port is `9003` by default. You can change the port.

5. Configure the `jetty` element to add a `secure-network-io-name` child element.

   The following example shows the default `jetty` element.

```
<jetty>
    <name>JettyServer</name>
    <network-io-name>NetIO</network-io-name>
    <work-manager-name>JettyWorkManager</work-manager-name>
    <secure-network-io-name>sslNetIo</secure-network-io-name>
</jetty>
```

   The `secure-network-io-name` must match the SSL `netio` element `name` child element.

6. Save and close the `config.xml` file.

7. Restart the Oracle Event Processing server (if it is running).

   See Start and Stop Servers.

9.5.2 Create a Key Store Manually

By default, the Configuration Wizard creates a default key store certificate file called `evsidentity.jks`.

By default the `evsidentity.jks` file is in the `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/ssl` directory.

The password is the same as the password you entered when you created the server. Optionally, you can create a key store manually.

1. Use the JDK `keytool` command to generate a key store:

```
keytool -genkey -alias evsidentity -keyalg RSA -validity 10958 -keystore
evsidentity.jks -keysize 1024
```

2. Enter the key store password, when prompted:

```
Enter keystore password:
```

3. Enter the key-store attributes, when prompted:

```
What is your first and last name?
  [Unknown]:  OEP
```

```
What is the name of your organizational unit?
  [Unknown]:  SOA
What is the name of your organization?
  [Unknown]:  ORACLE
What is the name of your City or Locality?
  [Unknown]:  SF
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=OEP, OU=SOA, O=ORACLE, L=SF, ST=CA, C=US correct?
  [no]:  y
```

4. When prompted for a key password, do not enter a password, but press **Return**:

```
Enter key password for <evsidentity>
        (RETURN if same as keystore password):
```

Note:

The Oracle Event Processing Server will not start unless the password for certificate private key is the same as the password for the identity key store.

---

5. In an XML editor, open the Oracle Event Processing server `config.xml` file.

   By default, the Configuration Wizard creates the `config.xml` file in the `/Oracle/Middleware/my_oep/user_projects/domains/` <domainname>/<*servername*>`/config` directory.

6. Configure the `ssl` element.

   The following example shows the default `ssl` element.

```
<ssl>
    <name>sslConfig</name>
    <key-store>KEYSTORE_PATH</key-store>
    <key-store-pass>
        <password>PASSWORD</password>
    </key-store-pass>
    <key-store-alias>KEYSTORE_ALIAS</key-store-alias>
    <key-manager-algorithm>SunX509</key-manager-algorithm>
    <ssl-protocol>TLS</ssl-protocol>
    <enforce-fips>false</enforce-fips>
    <need-client-auth>false</need-client-auth>
</ssl>
```

   KEYSTORE_PATH: The file path to the key store file. The file name comes from the `-keystore` argument to the `keytool` command.

   PASSWORD: The cleartext key store password.

   KEYSTORE_ALIAS: The key store alias. The key store alias is from the `-alias` argument to the `keytool` command.

7. Save and close the `config.xml` file.

8. Encrypt the cleartext password in the `key-store-pass` element `password` child element of the `config.xml` file by using the `encryptMSAConfig` utility.

   See The encryptMSAConfig Command-Line Utility.

9.5.3 Configure SSL in a Multiserver Domain for Visualizer

In a multiserver domain, you can configure one-way SSL between the server that hosts the Oracle Event Processing Visualizer data services application and another server. In the following procedure, the server that hosts the data services application is `myServer1`, and the second server is `myServer2`. Both servers are in the `/Oracle/Middleware/my_oep/user_projects/domains/myServer1` directory. Repeat this procedure for other servers in the domain, if required.

For information about securing the messages sent between servers in a multiserver domain, see:

- Oracle Coherence: Secure the Messages Sent Between Servers
- Native Clustering: Secure the Messages Sent Between Servers in a Multiserver Domain

For information about starting Oracle Event Processing Visualizer in a multiserver domain, see *Using Visualizer for Oracle Stream Explorer*.

Configure SSL in a Multiserver Domain for Use by Visualizer

1. Ensure that SSL is configured for the two servers in the domain.

   If you used the Configuration Wizard to create the servers, then SSL is configured by default.

   See Configure SSL Manually for details and information about how to change the default configuration.

2. Start `myServer2`.

   See Start and Stop Servers.

3. Change to the `ssl` sub-directory of the `myServer1` directory:

   ```
   cd /Oracle/Middleware/my_oep/user_projects/domains/myDomain/myServer1/ssl
   ```

4. Generate a trust key store for `myServer1` that includes the certificate of `myServer2` by specifying the following command (split for readability; in practice, the command should be on one line):

   ```
   prompt> java -classpath Oracle\Middleware\my_oep\oep\
   \common\lib\evspath.jar;Oracle\Middleware\my_oep\oep\utils\security
   \wlevsgrabcert.jar
   com.bea.wlevs.security.util.GrabCert host:secureport
   -alias=alias truststorepath
   ```

   `host`: The computer on which `myServer2` is running.

   `secureport`: The SSL network I/O port configured for `myServer2`. The default value is 9003. For more information, see Configure SSL Manually.

   `alias`: The alias for the certificate in the trust key store. Default value is the host name.

   `truststorepath`: The full path name of the generated trust key store file; default is `evstrust.jks`

   For example (put everything on one line):

```
java -classpath C:\Oracle\Middleware\
my_oep\oep\common\lib\evspath.jar;C:\Oracle\Middleware\
my_oep\oep\utils\security\wlevsgrabcert.jar
com.bea.wlevs.security.util.GrabCert myServer2:9003
-alias=myServer2 evstrust.jks
```

For more information, see The GrabCert Command-Line Utility.

5. When prompted, enter the Oracle Event Processing administrator password:

```
Please enter the Password for the trust store :
```

6. When prompted, select the certificate sent by `myServer2`:

```
Created TrustStore evstrust.jks
Opening connection to myServer2:9003...
Starting SSL handshake...

No certificates in evstrust.jks are trusted by myServer2:9003

Server sent 1 certificate(s):

 1 Subject CN=localhost, OU=Event Server, O=BEA, L=San Jose, ST=California, C=US
   Issuer  CN=localhost, OU=Event Server, O=BEA, L=San Jose, ST=California, C=US
   sha1    00 07 c0 f4 10 48 9a f9 07 82 4f b6 9c 7f 7c d0 37 57 90 7d
   md5     a4 d4 ff d2 43 69 95 ca c3 43 e6 f6 b8 08 df b7

Enter certificate to add to trusted keystore evstrust.jks or 'q' to quit: [1]
```

7. Update the `config.xml` file of `myServer1`, by adding trust key store information to the `ssl` element and adding a `use-secure-connections` element, as shown in bold in the following example:

```
<ssl>
    <name>sslConfig</name>
    <key-store>./ssl/evsidentity.jks</key-store>
    <key-store-pass>
        <password>{Salted-3DES}s4YUEvH4Wl2DAjb45iJnrw==</password>
    </key-store-pass>
    <key-store-alias>evsidentity</key-store-alias>
    <key-manager-algorithm>SunX509</key-manager-algorithm>
    <ssl-protocol>TLS</ssl-protocol>
    <trust-store>./ssl/evstrust.jks</trust-store>
    <trust-store-pass>
        <password>wlevs</password>
    </trust-store-pass>
    <trust-store-alias>evstrust</trust-store-alias>
    <trust-store-type>JKS</trust-store-type>
    <trust-manager-algorithm>SunX509</trust-manager-algorithm>
    <enforce-fips>false</enforce-fips>
    <need-client-auth>false</need-client-auth>
</ssl>
<use-secure-connections>
    <value>true</value>
</use-secure-connections>
```

The `config.xml` file is in the `config` directory of the main server directory. In this example, the location is `/Oracle/Middleware/my_oep/user_projects/domains/myDomain/myServer1/config/`.

8. Encrypt the cleartext password in the `trust-store-pass` element `password` child element of the `config.xml` file by using the `encryptMSAConfig` utility.

   See The encryptMSAConfig Command-Line Utility.

9. Start `myServer1`.

9.5.4 Configure SSL Between an SAML2 Service Provider and Identity Provider

You can use SSL to secure communications between the service provider (SP) and identity provider (IDP) in an SSO environment by configuring an Oracle Event Processing Server as a SAML2 SP and configure the SP with the needed SAML2 identity IDP options.

The following examples shows an `ssl` element with a sample configuration. The procedure edits the example to configure SSL between a SAML2 SP and an IP.

```
<ssl>
    <name>samlsslConfig</name>
    <key-store>security_files/DemoIdentity.jks</key-store>
    <key-store-pass>
        <password>DemoIdentityKeyStorePassPhrase</password>
    </key-store-pass>
    <key-store-type>JKS</key-store-type>
    <trust-store>security_files/DemoTrust.jks</trust-store>
    <ssl-protocol>TLS</ssl-protocol>
    <key-store-alias>demoidentity</key-store-alias>
    <key-manager-algorithm>SunX509</key-manager-algorithm>
    <enforce-fips>false</enforce-fips>
    <need-client-auth>false</need-client-auth>
</ssl>
```

Configure SSL between a SAML2 SP and an IP

1. Configure the mandatory SAML2 IDP options.

   See Configure SAML2 Identity Provider Options.

2. Add a `transport-layer-client-cert-alias` element to the `saml2-identity-provider` element in your Oracle Event Processing server `config.xml`:

   ```
   <transport-layer-client-cert-alias>sp1</transport-layer-client-cert-alias>
   ```

3. Add an `ssl-config-bean-name` element to the `saml2-identity-provider` element in your Oracle Event Processing server `config.xml`:

   ```
   <ssl-config-bean-name>samlsslConfig</ssl-config-bean-name>
   ```

   `ssl-config-bean-name`: The name of your ssl element.

4. Restart the Oracle Event Processing server for the changes to take effect.

   See "Start and Stop Servers".

9.6 FIPS

The National Institute of Standards and Technology (NIST) creates standards for Federal computer systems. NIST issues these standards as Federal Information Processing Standards (FIPS) for government-wide use.

Oracle Event Processing supports FIPS with the `com.rsa.jsafe.provider.JsafeJCE` security provider. Use this provider to configure Oracle Event Processing to use a FIPS-certified pseudo-random number generator for SSL.

For more information, see:

- SSL to Secure Network Traffic
- `http://www.itl.nist.gov/fipspubs/`

You can configure Oracle Event Processing servers to use a FIPS-certified pseudo-random number generator.

Configure FIPS for an Oracle Event Processing Server

1. Configure Java SE security.

   See Java SE Security for an Oracle Event Processing Server.

2. Configure SSL.

   See SSL to Secure Network Traffic.

3. Copy `com.bea.core.jsafejcefips_version.jar`:

   From: */Oracle/Middleware/my_oep/oep/utils/security*

   To: *JRE_HOME*`/jre/lib/ext`

   JRE_HOME : The directory that contains your JDK installation.

4. Stop the Oracle Event Processing server, if it is currently running.

   See Start and Stop Servers.

5. Edit the *JRE_HOME*`/jre/lib/security/java.security file to add com.bea.core.jsafejcefips_2.0.0.0.jar` as a JCE provider as the following example shows.

   ```
   security.provider.N=com.rsa.jsafe.provider.JsafeJCE
   ```

   N: A unique integer that specifies the order in which Java accesses security providers. To make the `JsafeJCE` provider the default provider, set *N* to 1. In this case, change the value of *N* for any other providers in the `java.security` file so that each provider has a unique number as the following shows.

   ```
   security.provider.1=com.rsa.jsafe.provider.JsafeJCE
   security.provider.2=sun.security.provider.Sun
   ```

6. Edit the `ssl` element in the `config.xml` server file to add the following child elements:

   - `enforce-fips`: set this option to `true`.
   - `secure-random-algorithm`: set this option to FIPS186PRNG
   - `secure-random-provider`: set this option to `JsafeJCE`.

   ```
   <ssl>
       <name>sslConfig</name>
       <key-store>./ssl/evsidentity.jks</key-store>
   ```

```
        <key-store-pass>
            <password>s4YUEvH4Wl2DAjb45iJnrw==</password>
        </key-store-pass>
        <key-store-alias>evsidentity</key-store-alias>
        <key-manager-algorithm>SunX509</key-manager-algorithm>
        <ssl-protocol>TLS</ssl-protocol>
        <enforce-fips>true</enforce-fips>
        <need-client-auth>false</need-client-auth>
        <secure-random-algorithm>FIPS186PRNG</secure-random-algorithm>
        <secure-random-provider>JsafeJCE</secure-random-provider>
    </ssl>
```

7. Restart the Oracle Event Processing server for the changes to take effect.

   See Start and Stop Servers.

9.7 SSO with SAML2

The Security Assertion Markup Language (SAML) is an OASIS XML standard for exchanging authentication and authorization data between security domains. Oracle Event Processing server supports SAML2.

With SAML configuration, you can define a web application single sign-on (SSO) environment between Oracle Event Processing servers and a SAML-compliant system such as Oracle WebLogic Server or Oracle Access Manager.

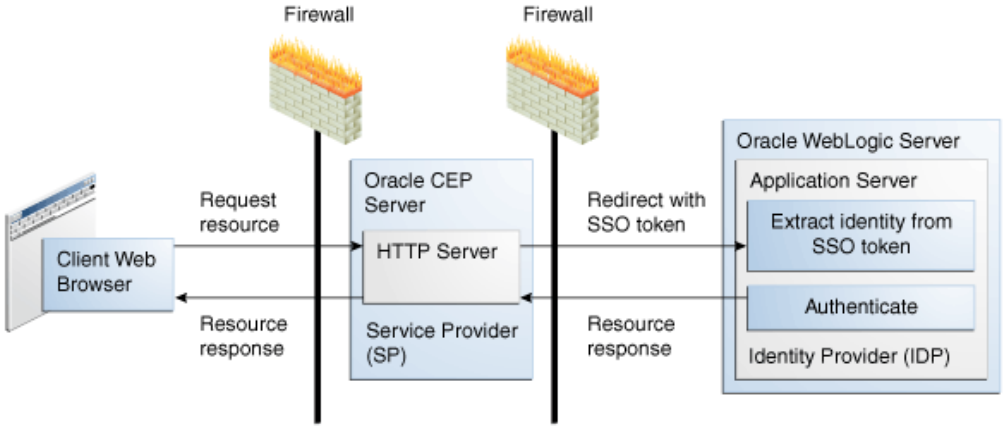
SSO enables you to have a user to sign on to an application once and gain access to many different application components even when these components have their own authentication schemes. Single sign-on enables users to log in securely to all of their applications with one identity.

There are two roles in a SAML2 SSO environment as Figure 9-1 shows:

- Identity Provider (IDP): The process that performs authentication. You configure SAML2 options for the IDP in the Oracle Event Processing server `config.xml` file.
- Service Provider (SP): The process that delegates authentication to an IDP. You configure SAML2 options for the SP in the Oracle Event Processing `security.xml` file.

In the context of Oracle Event Processing, the Oracle Event Processing server is the service provider. The identity provider is any SAML2-compliant system such as Oracle WebLogic Server or Oracle Access Manager.

Figure 9-1 SSO Using SAML2



Be aware that Oracle Event Processing Visualizer supports SSO with SAML2, but the Oracle Event Processing HTTP Publish-Subscribe Server (HTTP pub-sub server) does *not* support SSO with SAML2.

For more information, see:

- *Understanding Security for Oracle WebLogic Server*
- *Developing Applications with the WebLogic Security Service*
- `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`.

The following procedures explain how to configure SSO with SAML2 on an Oracle Event Processing server:

- Configure SAML2 Service Provider Options
- Configure SAML2 Identity Provider Options
- Configure SAML2 Web Application Options.

9.7.1 Configure SAML2 Service Provider Options

In this configuration, the Oracle Event Processing server receives client requests and delegates their authentication to a SAML2 IDP. You configure SAML2 SP options in the `security.xml` file with the `cssconfig` command-line tool.

Configure SAML2 Service Provider Options

1. Add the *`Oracle\Middleware\my_oep\oep\bin`* directory to your `PATH` environment variable:

   ```
   set PATH=d:\Oracle\Middleware\my_oep\oep\bin;%PATH% (Windows)
   PATH=/Oracle/Middleware/my_oep/oep/bin:$PATH (UNIX)
   ```

2. Change to the `config` directory for the server you want to update.

   By default the domain directory is `Oracle/Middleware/my_oep/user_projects/domains/<domainname>`.

3. Make a backup copy of the existing `security.xml` file in case you need to revert:

4. In a text editor, create a file called `saml2.properties` and copy the entire contents of the following example into it.

   You can customize the properties file.

   ```
   samlkey.mbean=com.bea.common.management.configuration.SAMLKeyServiceMBean
   samlkey.Type=JKS
   samlkey.Filename=security_files\\DemoIdentity.jks
   samlkey.Passphrase=keystore_passphrase
   samlkey.DefaultAlias=demoidentity
   samlkey.DefaultPassphrase=keystore_passphrase
   saml2.mbean=com.bea.common.management.configuration.SingleSignOnServicesMBean
   saml2.PublishedSiteURL=http://localhost:9002/saml2
   saml2.EntityID=http://localhost:9002/saml2
   saml2.ServiceProviderEnabled=true
   saml2.ServiceProviderPreferredBinding=HTTP/POST
   saml2.SignAuthnRequests=true
   saml2.WantAssertionsSigned=true
   saml2.SSOSigningKeyAlias=sp1
   saml2.SSOSigningKeyPassPhrase=password
   ```

```
saml2.TransportLayerSecurityKeyAlias=sp1
saml2.TransportLayerSecurityKeyPassPhrase=password
saml2.BasicAuthUsername=spBasicAuth
saml2.BasicAuthPassword=password
saml2.WantArtifactRequestsSigned=true
saml2.WantTransportLayerSecurityClientAuthentication=false
atn.10.mbean=weblogic.security.providers.saml.SAMLAuthenticatorMBean
atn.10.Name=saml2-atn-provider
ia.4.mbean=com.bea.security.saml2.providers.SAML2IdentityAsserterMBean
ia.4.Name=saml2-Identity-Asserter
```

5. Create a new security configuration file (`security.xml`) by executing the following `cssconfig` command:

   ```
   cssconfig -p saml2.properties -i security-key.dat -c security.xml
   ```

   `saml2.properties`: The property file you created in `security.xml` is the name of the new security configuration file, and `security-key.dat`: An existing file, generated by the Configuration Wizard, that contains the identity key.

   See The cssconfig Command-Line Utility for additional information.

6. Restart the Oracle Event Processing server for the changes to take effect.

   See "Start and Stop Servers".

9.7.2 Configure SAML2 Identity Provider Options

In this configuration, you configure an Oracle Event Processing server with SAML2 IDP options that the server then uses to delegate authentication to a SAML2-compliant IDP. You configure SAML2 IDP options in the Oracle Event Processing server `config.xml` file.

This procedure uses Oracle WebLogic Server as an example IDP. Refer to your IDP documentation for configuration details specific to your IDP and use this procedure as a guide.

Configure SAML2 Identity Provider Options

1. Obtain the IDP metadata file from your IDP.

   If Oracle WebLogic Server is your IDP, you can generate the IDP metadata file as follows:

   a. Open a browser and log into the Oracle WebLogic Server console:

      ```
      http://localhost:1010/console
      ```

   b. Under your domain, select **Environment > Servers > *SERVER_NAME* > Federation > Services > SAML 2.0 General**

      Where *SERVER_NAME* is the name of your Oracle WebLogic Server.

   c. Click **Publish Meta Data** and specify a file name.

      In this example, call it `myidp.xml`.

2. Change to the `config` directory for the server you want to update.

   By default, the directory is `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config`.

3. Copy the `myidp.xml` file to your Oracle Event Processing server `config` directory.
4. Make a backup copy of the existing `config.xml` file, in case you need to revert:
5. In a text editor, edit the `config.xml` file and add a `saml2-identity-provider` element as the following example shows.

   You can configure only one IDP per Oracle Event Processing server.

   ```
   <saml2-identity-provider>
       <meta-data-file-name>myidp.xml</meta-data-file-name>
       <partner-name>partnerIdP1</partner-name>
       <redirect-uris>
           <uri>/unleashed_saml/</uri>
           <uri>/unheardof_saml/</uri>
       </redirect-uris>
   </saml2-identity-provider>
   ```

   `meta-data-file-name`: `partner-name`: The name of this IDP instance.`redirect-uris`: SAML2 authentication URIs.
6. Restart the Oracle Event Processing server for the changes to take effect.

   See Start and Stop Servers.

9.7.3 Configure SAML2 Web Application Options

After you configure SAML2 SP and IDP options, you must configure the `web.xml` file for the web applications that will access the SP.

Configure SAML2 Web Application Options

1. Change to the directory that contains the `web.xml` file for the web application that needs to access the SP.
2. In a text editor, edit the `web.xml` file and add a `filter` element:

   ```
   <filter>
     <filter-name>SAML2Filter</filter-name>
     <filter-class>com.bea.core.saml2.SAML2Filter</filter-class>
   </filter>
   ```
3. Add a `filter-mapping` element:

   ```
   <filter-mapping>
     <filter-name>SAML2Filter</filter-name>
     <url-pattern>/welcome.jsp</url-pattern>
   </filter-mapping>
   ```

   The protected resource defined in the `filter-mapping` child element `url-pattern` must match the `redirectUri` you configured in the `config.xml` file.
4. Repackage and deploy the web application.

9.8 HTTPS-Only Connections

You can lock down the server to allow only HTTPS connections.

Configure HTTPS-Only Connections for a Server

1. Ensure that SSL is configured for the server.

   See SSL to Secure Network Traffic for details.

2. Remove the HTTP port configuration from the config.xml server file.

   By default the file is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config/`

   The following example shows `config.xml` entries with a standard configuration where both an HTTP and HTTPS ports are configured. The HTTP port is 9002 and the HTTPS port is 9003. Clients can access the Jetty server through both ports.

```
<netio>
   <name>NetIO</name>
   <port>9002</port>
</netio>
<netio>
   <name>sslNetIo</name>
   <port>9003</port>
   <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
</netio>
<jetty>
   <name>JettyServer</name>
   <network-io-name>NetIO</network-io-name>
   <secure-network-io-name>sslNetIo</secure-network-io-name>
   ...
</jetty>
<ssl>
   <name>sslConfig</name>
   <key-store>./ssl/evsidentity.jks</key-store>
   ...
</ssl>
```

   The following example shows the same `config.xml` file with HTTP access removed. Clients can now access the Jetty server using the HTTPS port only.

```
<netio>
   <name>sslNetIo</name>
   <port>9003</port>
   <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
</netio>
<jetty>
   <name>JettyServer</name>
   <secure-network-io-name>sslNetIo</secure-network-io-name>
   ...
</jetty>
<ssl>
   <name>sslConfig</name>
   <key-store>./ssl/evsidentity.jks</key-store>
   ...
</ssl>
```

3. If you have a multiserver domain, make sure that SSL is configured between the member servers.

   See Configure SSL in a Multiserver Domain for Visualizer for details.

9.9 Security for Server Services

After you complete basic security tasks such as configuring Java SE security, a security service provider, and SSL, you can configure security details specific to the various services that Oracle Event Processing server provides.

This section describes:

- Configure Jetty Security
- Configure JMX Security
- Configure JDBC Security
- Configure HTTP Publish-Subscribe Server Channel Security

9.9.1 Configure Jetty Security

Oracle Event Processing supports Jetty as a Java web server to deploy HTTP servlets and static resources. See `http://mvnrepository.com/artifact/org.mortbay.jetty`. For more information about Jetty, see Jetty.

The following security tasks affect Jetty configuration:

- Java SE Security for an Oracle Event Processing Server
- SSL to Secure Network Traffic.

9.9.2 Configure JMX Security

Clients that access the Oracle Event Processing server with JMX are subject to Oracle Event Processing role-based authentication. For more information about roles, see:

- Users, Groups, and Roles
- *Using Visualizer for Oracle Stream Explorer*.

For more information about JMX, see JMX .

9.9.3 Configure JDBC Security

If you update a data source with a new password using the Configuration Wizard, then the Configuration Wizard performs password encryption for you. If you update the `config.xml` file manually by adding or modifying a `data-source` element, then you enter the password in plain text and encrypt the password with the encryption utility, `encryptMSAConfig`.

The following example shows a `config.xml` file `data-source` element with a new plain text password, `secret`, specified in the `properties` element with the name `password`.

```
<data-source>
   <name>epcisDS</name>
   <driver-params>
      <url>jdbc:sqlserver://localhost:1433;databaseName=myDB;SelectMethod=cursor</url>
      <driver-name>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-name>
      <properties>
         <element>
            <name>user</name>
            <value>juliet</value>
         </element>
```

```
         <element>
            <name>password</name>
            <value>secret</value>
         </element>
      </properties>
   </driver-params>
</data-source>
<transaction-manager>
   <name>TM</name>
   <rmi-service-name>RMI</rmi-service-name>
</transaction-manager>
```

The following example shows the `config.xml` file `data-source` element after encryption. Note the plain text password is encrypted.

```
<data-source>
   <name>epcisDS</name>
   <driver-params>
      <url>jdbc:sqlserver://localhost:1433;databaseName=myDB;SelectMethod=cursor</url>
      <driver-name>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-name>
      <properties>
         <element>
            <name>user</name>
            <value>juliet</value>
         </element>
         <element>
            <name>password</name>
            <value>{Salted-3DES}hVgC5iZ3nZA=</value>
         </element>
      </properties>
   </driver-params>
</data-source>
<transaction-manager>
   <name>TM</name>
   <rmi-service-name>RMI</rmi-service-name>
</transaction-manager>
```

For more information, see:

- User Credentials for Command-Line Utilities
- The encryptMSAConfig Command-Line Utility.

For more information about JDBC, see JDBC.

9.9.4 Configure HTTP Publish-Subscribe Server Channel Security

After you configure at least one HTTP publish-subscribe server channel, you can use role-based authentication to control access to individual HTTP publish-subscribe server channels using the Oracle Event Processing Visualizer.

For more information, see:

- Users, Groups, and Roles
- HTTP Publish-Subscribe Server
- *Using Visualizer for Oracle Stream Explorer*.

9.10 Cross-Domain Security for Visualizer

Oracle Event Processing Visualizer provides an Adobe Flash-based user interface with which you can create and configure event processing networks. To provide the most flexible default performance for Visualizer, the software is installed with a configured

trust level that allows access to Visualizer data from any domain. If this trust level is inappropriate for your deployment, you can edit the application's Flash cross-domain policy file to restrict access.

Review the domains the Flash cross-domain policy allows and determine whether it is appropriate for the application to fully trust both the intentions and security posture of those domains. See the Adobe web site for a thorough description of editing cross-domain policy. See the Adobe security web site for information about using the Adobe cross-domain policy files.

Update Cross-Domain Security

1. Locate the Oracle Event Processing Visualizer JAR file.

   By default the file is in `Oracle/Middleware/my_oep/oep/modules/com.bea.wlevs.visualizer.jmxhttpadapter_version.jar`.

2. Expand the JAR file and locate the `crossdomain.war` file.

3. Expand the `crossdomain.war` file to locate the `crossdomain.xml` file.

4. Edit the `crossdomain.xml` file to reflect your cross-domain security needs.

5. Repackage the `crossdomain.war` file and the Oracle Event Processing Visualizer JAR file.

9.11 Security Auditor

Oracle Event Processing provides a security auditor that logs security-related activity. By default, the security auditor logs to

```
Oracle/Middleware/my_oep/user_projects/domains/<domainname>/ <servername>/legacy-rootdir/servers/legacy-server-name/logs
```

By default, the Oracle Event Processing security auditor logs security errors or failures to keep the security auditor log file at a manageable size. You can configure the level at which the Oracle Event Processing security auditor logs information.

For more information, see *Administering Security for Oracle WebLogic Server 12c (12.2.1)*.

To configure security auditor logging:

1. Change to the `config` directory for the server you want to configure.

2. In a text editor, open the `security.xml` file and locate the `sec:auditor` element element:

   ```
   <sec:auditor xsi:type="wls:default-auditorType">
     <sec:name>my-auditor</sec:name>
     <wls:severity>CUSTOM</wls:severity>
     <wls:rotation-minutes>720</wls:rotation-minutes>
     <wls:error-audit-severity-enabled>true</wls:error-audit-severity-enabled>
     <wls:failure-audit-severity-enabled>true</wls:failure-audit-severity-enabled>
   </sec:auditor>
   ```

3. Modify the `sec:auditor` element as required:

   `wls:rotation-minutes`: How many minutes to wait before creating a new `DefaultAuditRecorder.log` file. At the specified time, the audit file closes and a new files is created. Oracle Event Processing creates a backup file named `DefaultAuditRecorder.YYYYMMDDHHMM.log` in the same directory.

`wls:severity`: The severity level from the following list that is appropriate for your server. The security auditor audits security events of the specified severity and all events with a higher numeric severity rank. For example, if you set the severity level to `ERROR`, the Oracle Event Processing security auditor audits security events of severity level `ERROR`, `SUCCESS`, and `FAILURE`.

`INFORMATION`: `1WARNING`: `2ERROR`: `3SUCCESS`: `4FAILURE`: `5`

You can also set the `wls:severity` level to `CUSTOM`, and enable (`true`) or disable (`false`) the specific severity levels you want to audit by using one or more of the following child elements:

- `wls:information-audit-severity-enabled`: If the `severity` value is set to `CUSTOM`, setting this child element to true causes the Oracle Event Processing security auditor to generate audit records for events with a severity level of `INFORMATION`.
- `wls:warning-audit-severity-enabled`: If the `severity` value is set to `CUSTOM`, setting this child element to true causes the Oracle Event Processing security auditor to generate audit records for events with a severity level of `WARNING`.
- `wls:error-audit-severity-enabled`: If the `severity` value is set to `CUSTOM`, setting this child element to true causes the Oracle Event Processing security auditor to generate audit records for events with a severity level of `ERROR`.
- `wls:success-audit-severity-enabled`: If the `severity` value is set to `CUSTOM`, setting this child element to true causes the Oracle Event Processing security auditor to generate audit records for events with a severity level of `SUCCESS`.
- `wls:failure-audit-severity-enabled`: If the `severity` value is set to `CUSTOM`, setting this child element to true causes the Oracle Event Processing security auditor to generate audit records for events with a severity level of `FAILURE`.

4. Save and close the `security.xml` file.
5. Restart the Oracle Event Processing server for the changes to take effect.

   See Start and Stop Servers.

9.12 Disable Security

You can disable security entirely on the Oracle Event Processing server. While this configuration might be appropriate for development environments, Oracle does not recommend disabling security in a production environment.

To temporarily disable security, you can run the `startwlevs.cmd` or `startwlevs.sh` script with the `-disablesecurity` argument on the command line. For example:

```
startwlevs.cmd -disablesecurity
```

Note:

In some sample domains, the `startwlevs.cmd` and `startwlevs.sh` scripts already include a `-disablesecurity` argument. Executing such a script with `-disablesecurity` on the command line will fail with an `Illegal argument` error.

9.13 Security Utilities

Oracle Event Processing provides a variety of command-line utilities to simplify security administration. In addition to command-line utilities, you can use Oracle Event Processing Visualizer to perform many security tasks.

For more information, see:

- Security Utilities Command-Line Reference
- User Credentials for Command-Line Utilities
- *Using Visualizer for Oracle Stream Explorer*.

9.14 User Credentials for Command-Line Utilities

Oracle Event Processing provides the following command-line utilities for performing a variety of tasks:

- `wlevs.Admin`: a command-line interface to administer Oracle Event Processing and, in particular, dynamically configure the rules for Oracle CQ processors and monitor the event latency and throughput of an application. See wlevs.Admin Command-Line Reference for details
- `Deployer`: a Java-based deployment utility that provides administrators and developers command-line based operations for deploying Oracle Event Processing applications. See Deployer Command-Line Reference for details.
- `cssconfig`: a command-line utility to generate a security configuration file (`security.xml`) that uses a password policy. See The cssconfig Command-Line Utility for details.
- `encryptMSAConfig`: an encryption command-line utility to encrypt cleartext passwords, specified by the `password` element, in XML files. See The encryptMSAConfig Command-Line Utility for details.

For each utility, you can specify user credentials (user name and password) using the following three methods:

- On the command line using options such as `-user` and `-password`.
- Interactively so that the command line utility always prompts for the credentials.
- Specifying a filestore that stores the user credentials; the filestore itself is also password protected.

In a production environment you should *never* use the first option (specifying user credentials on the command line) but rather use only the second and third option.

When using interactive mode (command-line utility prompts for credentials), be sure you have the appropriate `terminalio` native libraries for your local computer in

your `CLASSPATH` so that the user credentials are not echoed on the screen when you type them. Oracle Event Processing includes a set of standard native libraries for this purpose, but it may not include the specific one you need.

9.15 Security in Oracle Event Processing Examples and Domains

When you use the Configuration Wizard to create a new domain, you specify the administrator user and password, as well as the password to the domain identity key store. This user is automatically added to the `wlevsAdministrators` group. All security configuration is stored using a file-based provider, by default.

All Oracle Event Processing examples are configured to have an administrator with user name `oepadmin` and password `welcome1`. When you create a new domain you specify the administrator name and password.

By default, security is disabled in the HelloWorld example. This means that any user can start the server, deploy applications, and run all commands of the administration tool (`wlevs.Admin`) without providing a password.

Security is enabled in the FX and AlgoTrading examples. In both examples, the user `oepadmin`, with password `welcome1`, is configured to be the Oracle Event Processing administrator with full administrator privileges. The scripts to start the server for these examples use the appropriate arguments to pass this user name and password to the `java` command. If you use the Deployer or `wlevs.Admin` utility, you must also pass this user name/password pair using the appropriate arguments.

For more information, see User Credentials for Command-Line Utilities.

10

Jetty

Oracle Event Processing supports Jetty as a Java web server to deploy HTTP servlets and static resources. You can configure Jetty features to use them with Oracle Event Processing. Features you can configure are network I/O, work managers, and configuring a Jetty server instance.

Oracle Event Processing Jetty support is based on Version 1.2 the OSGi HTTP Service. The OSGi HTTP Service API enables dynamically registering and unregistering objects with run time and static resources. This specification requires Java Servlet API 2.1 or higher. See `http://java.sun.com/products/servlet/docs.html`.

This chapter includes the following sections:

10.1 Jetty Features

Oracle Event Processing supports the following Jetty features:

Servlets: Oracle Event Processing supports synchronous and asynchronous Java servlets. An asynchronous servlet receives a request, gets a thread and performs some work, and finally releases the thread while waiting for those actions to complete before re-acquiring another thread and sending a response. See Application Development and Deployment.

Network I/O Integration: Oracle Event Processing uses network I/O (Net IO) to configure the port and listen address of Jetty services. Jetty has a built-in capability for multiplexed network I/O. However, it does not support multiple protocols on the same port.

Thread Pool Integration: Oracle Event Processing Jetty services use the Oracle Event Processing work manager for scalable thread pooling. See Example Jetty Configuration. Although, Jetty provides its own thread pooling capability, Oracle recommends that you use the Oracle Event Processing self-tuning thread pool to minimize footprint and configuration complexity.

Jetty Work Managers: Oracle Event Processing enables you to configure how your application prioritizes the execution of its work. You define rules and monitor run time performance to optimize application performance and maintain service-level agreements. You define a work manager to define the rules and constraints for your application. See Jetty Configuration Objects.

10.2 Thread Pools

Oracle Event Processing uses a single thread pool to execute all types of work. Oracle Event Processing prioritizes work based on rules you define and run-time metrics that include the time it takes to execute a request, and the rate at which requests enter and leave the pool.

The common thread pool changes size to maximize throughput. The queue monitors throughput over time, and based on history, determines whether to adjust the thread count. For example, if historical throughput statistics indicate that a higher thread count increased throughput, Oracle Event Processing increases the thread count. Similarly, if statistics indicate that fewer threads did not reduce throughput, Oracle Event Processing decreases the thread count.

10.3 Work Manager Configuration

Oracle Event Processing prioritizes work and allocates threads based on an execution model that accounts for defined parameters, and run time performance and throughput. You can configure a set of scheduling guidelines and associate them with one or more applications or with particular application components.

For example, you can associate one set of scheduling guidelines for one application, and another set of guidelines for other applications. At run time, Oracle Event Processing uses the guidelines to assign pending work and enqueued requests to execution threads.

To manage work in your applications, define one or more of the following work manager components:

- `fairshare`: The average thread-use time required to process requests.

  For example, Oracle Event Processing runs two modules and the work manager for `ModuleA` specifies a `fairshare` of 80 and the work manager for `ModuleB` specifies a `fairshare` of 20. During a period of sufficient demand, with a steady stream of requests for each module such that the number requests exceed the number of threads, Oracle Event Processing allocates 80% and 20% of the thread-usage time to `ModuleA` and `ModuleB`, respectively.

  Note:

  You specify a fair share request class as a relative value, not a percentage. Therefore, in the above example, if the request classes are defined as 400 and 100, they still have the same relative values.

- `max-threads-constraint`: Limits the number of concurrent threads executing requests from the constrained work set. The default is unlimited. For example, consider a constraint defined with maximum threads of 10 and shared by 3 entry points. The scheduling logic ensures that not more than 10 threads are executing requests from the three entry points combined.

  A `max-threads-constraint` can be defined in terms of a the availability of resource that requests depend upon, such as a connection pool.

  A `max-threads-constraint` might, but does not necessarily, prevent a request class from taking its fair share of threads or meeting its response time goal. Once the constraint is reached the Oracle Event Processing does not schedule requests of

this type until the number of concurrent executions falls below the limit. The Oracle Event Processing then schedules work based on the fair share or response time goal.

- `min-threads-constraint`: Guarantees a number of threads the server will allocate to affected requests to avoid deadlocks. The default is zero. A `min-threads-constraint` value of one is useful, for example, for a replication update request, which is called synchronously from a peer.

  A `min-threads-constraint` might not increase a fair share. This type of constraint has an effect primarily when the Oracle Event Processing instance is close to a deadlock condition. In that case, the constraint causes Oracle Event Processing to schedule a request even when requests in the service class have gotten more than their fair share recently.

10.4 Application Development and Deployment

Oracle Event Processing supports servlet development for deployment to Jetty.

The developer creates a standard Java EE web application and configures it with the `jetty-web-app` configuration object in the `config.xml` file for the server where the servlet will run.

Oracle Event Processing supports servlet deployments packaged either as war files or as exploded war files, as described in version 2.4 of the Java Servlet Specification. You can deploy pre-configured web applications from an exploded directory or war file by including them in the server configuration.

Security constraints specified in the standard web.xml file are mapped to the Common Security Services security provider. The Servlet API specifies declarative role-based security, which means that particular URL patterns can be mapped to security roles.

10.5 Configure a Jetty Server Instance

This section presents an example Jetty configuration followed by a description of the Jetty configuration objects.

You configure a Jetty server instance in the `config.xml` file for the server you want to configure.For information about security configuration tasks that affect Jetty, see Configure Jetty Security.

10.5.1 Example Jetty Configuration

The following snippet from a `config.xml` file provides an example Jetty configuration. Only Jetty-related configuration information is shown:

```
<config>
  <netio>
    <name>JettyNetIO</name>
    <port>9002</port>
  </netio>
  <work-manager>
    <name>WM</name>
    <max-threads-constraint>64</max-threads-constraint>
    <min-threads-constraint>3</min-threads-constraint>
  </work-manager>
  <jetty>
    <name>TestJetty</name>
    <work-manager-name>WM</work-manager-name>
    <network-io-name>JettyNetIO</network-io-name>
    <debug-enabled>false</debug-enabled>
```

```
    <scratch-directory>JettyWork</scratch-directory>
  </jetty>
  <jetty-web-app>
    <name>test</name>
    <context-path>/test</context-path>
    <path>testWebApp.war</path>
    <jetty-name>TestJetty</jetty-name>
  </jetty-web-app>
</config>
```

10.5.2 Jetty Configuration Objects

This section explains how to use the following configuration objects to configure an instance of the Jetty HTTP server. You configure a Jetty HTTP server in the `config.xml` file that describes your Oracle Event Processing server.

- jetty
- netio
- work-manager
- jetty-web-app.

jetty

Use the parameters described in the following table to define a `jetty` configuration object in your `config.xml` file.

Table 10-1 jetty Element Configuration Parameters

Parameter	Type	Description
`network-io-name`	`String`	The name of the Net IO service used. The Net IO service defines the port the server listens on. See Jetty Configuration Objects for details.
`work-manager-name`	`String`	The name of the Work Manager that should be used for thread pooling. If not specified, the default work manager is used. See Jetty Configuration Objects.
`scratch-directory`	`String`	The name of a directory where temporary files required for web applications, JSPs, and other types of web artifacts are kept.
`debug-enabled`	`boolean`	Enable debugging in the Jetty code using the OSGi Log Service.
`name`	`String`	The name of the jetty server instance.

netio

Use the parameters described in the following table to define a `netio` configuration object in your `config.xml` file.

Table 10-2 netio Element Configuration Parameters

Parameter	Type	Description
`name`	`String`	The name of the configuration object.
`port`	`int`	The listening port number.
`listen-address`	`String`	The address on which an instance of `netio` service listens for incoming connections. • It can be set to a numeric IP address in the a.b.c.d format, or to a host name. • If not set, the service listens on all network interfaces. The value of this parameter cannot be validated until the service has started.

work-manager

Use the parameters described in the following table to define a `work-manager` configuration object in your `config.xml` file.

Table 10-3 work-manager Element Configuration Parameters

Parameter	Type	Description
`min-threads-constraint`	`Integer`	The minimum threads this work manager uses.
`fairshare`	`Integer`	The `fairshare` value this work manager uses.
`max-threads-constraint`	`Integer`	The maximum threads constraint this work manager uses.
`name`	`String`	The name of this work manager.

jetty-web-app

Use the following configuration object to define a web application for use by Jetty.

Table 10-4 jetty-web-app Element Configuration Parameters

Parameter	Type	Description
`name`	`String`	The name of this work manager.
`context-path`	`String`	The context path where this web app is deployed in the web server's name space. If not set, it defaults to "`/`".
`scratch-directory`	`String`	The location where Jetty stores temporary files for this web app. Overrides the `scratch-directory` parameter in the Configure a Jetty Server Instance.

Parameter	Type	Description
`path`	`String`	A file name that points to the location of the web app on the server. It may be a directory or a war file.
`jetty-name`	`String`	The name of the Jetty service where this web application is deployed. It must match the name of an existing Configure a Jetty Server Instance.
`name`	`String`	The name of this configuration object.

11

JMX

Oracle Event Processing provides standards-based interfaces that are fully compliant with the Java Management Extensions (JMX) specification. Software developers can use these interfaces to monitor Oracle Event Processing management beans (MBeans), to change the configuration of an Oracle Event Processing domain, and to monitor Oracle Event Processing applications.

This chapter includes the following sections:

- MBean Usage
- Access the Oracle Event Processing JMX Server
- Types of MBeans
- Configure JMX
- Manage with JMX.

11.1 MBean Usage

Software developers implement MBean interfaces to design and develop an Oracle Event Processing management console to be used by the administrators at a customer installation. MBeans enable a developer to dynamically configure EPN components and perform server, domain, and application configuration and life cycle management. EPN configuration tasks include adding and removing Oracle CQL or rules, changing the channel maximum size, subscribing to notifications, and executing operations.

Currently there is no MBean support for deploying and undeploying application libraries on a local or remote server. See Deployer Command-Line Reference for more information.

You can manipulate MBeans with any of the following tools:

- Oracle Event Processing Visualizer. See *Using Visualizer for Oracle Stream Explorer*.
- `wlevs.Admin` command-line utility. See wlevs.Admin Command-Line Reference.
- Deployer command-line deployment utility. See Deployer Command-Line Reference.
- `jconsole`, which is the JMX console provided by the JDK.
- In Java code with standard JMX APIs. See http://www.oracle.com/technetwork/java/javase/tech/index-jsp-142926.html.

11.2 Access the Oracle Event Processing JMX Server

To access Oracle Event Processing MBeans, you must first connect to the Oracle Event Processing JMX server.

Oracle Event Processing does not support the JRMP protocol. Instead, JMX clients must use the more secure MSA protocol for both local and remote access to the Oracle Event Processing JMX server.

When you connect to the Oracle Event Processing JMX server that is running on `localhost` or on a remote host, you must copy the following Oracle Event Processing server JAR files to the client class path of the host from which you want to connect to the Oracle Event Processing server:

- *`Oracle\Middleware\my_oep\oep`*`\wlserver\modules\com.bea.core.jmx_13.0.0.0.jar`
- *`Oracle\Middleware\my_oep\oep\wlserver\modules\com.bea.core.rmi_13.0.0.0.jar`*
- *`Oracle\Middleware\my_oep\oep\wlserver\modules\com.bea.core.jndi.context_13.0.0.0.jar`*
- *`Oracle\Middleware\my_oep\oep\wlserver\modules\com.bea.core.logging_3.0.0.0.jar`*
- *`Oracle\Middleware\my_oep\oep\wlserver\modules\com.bea.core.bootbundle_13.0.0.0.jar`*

You must launch your JMX client (such as `jconsole`) using the following command line options and classpath (split for readability.

```
java -Djmx.remote.protocol.provider.pkgs=com.bea.core.jmx.remote.provider
-Dmx4j.remote.resolver.pkgs=com.bea.core.jmx.remote.resolver
-Djava.naming.factory.initial=com.bea.core.jndi.context.ContextFactory
-classpath %JAVA_HOME%\lib\jconsole.jar;MODULE_HOME\modules
\com.bea.core.jmx_13.0.0.0.jar;
MODULE_HOME\modules\com.bea.core.rmi_7.0.0.0.jar;
MODULE_HOME\modules\com.bea.core.jndi.context_7.0.0.0.jar;
MODULE_HOME\modules\com.bea.core.logging_1.5.0.0.jar;
MODULE_HOME\modules\com.bea.core.bootbundle_8.0.0.0.jar
sun.tools.jconsole.JConsole
```

Where *`MODULE_HOME`* is the directory you copied the Oracle Event Processing server JAR files to.

To connect to the Oracle Event Processing JMX server, you must use the JMX URL `service:jmx:msarmi://`*`HOST-NAME`*`:`*`port`*`/jndi/jmxconnector` so that you are always using the MSA connector (where *`HOST-NAME`* is either `localhost` or the name of the remote host and *`port`* is the Oracle Event Processing server JNDI port).

For more information, see:

- Connect with APIs to a JMX Server from a Non-Oracle Event Processing Client
- Connect to Local or Remote JMX Server Using JConsole with Security Disabled.

11.3 Types of MBeans

Oracle Event Processing exposes configuration and run time MBeans.

Configuration MBeans contain configuration information about EPN components, a deployed applications, servers, and domains. These MBeans have a fixed management interface and represent the information contained in the server config.xml file and the component configuration file. `CQLProcessorMBean` and `EventChannelMBean` are examples of configuration MBeans.

Every component in a deployed application (adapter, channel, or processor) has a configuration MBean that manages the underlying configuration of the component. Each type of component has its own set of manageable artifacts. For example, you can dynamically configure the maximum number of threads for a channel or the Oracle CQL rules associated with a processor.

Run time MBeans contain monitoring information for each component in the EPN with run time MBeans. Monitoring information includes throughput (number of events passing through a component) and latency (how long it takes an event to pass through a component).

See the API Overview in *Developing Applications for Event Processing with Oracle Stream Explorer* for a list of packages that contain MBeans. See *Java API Reference for Oracle Stream Explorer* for information about specific MBeans.

11.3.1 Configuration MBeans

When you deploy an Oracle Event Processing application, the server creates a configuration MBean for each component in the EPN whose manageability has been enabled or for each component registered in the EPN assembly file. If you have extended the configuration of an adapter, then the server deploys a custom configuration MBean for the adapter.

Using JMX, you can dynamically configure the component using its configuration MBean. For example, using the `EventChannelMBean.setMaxSize()` method you can set the size of a channel component.

11.3.2 Configuration MBean Naming

Oracle Event Processing configuration MBeans are arranged in a hierarchy. The object name of each MBean reflects its position in the hierarchy.

A typical object naming pattern is as follows:

```
com.bea.wlevs:Name=name,Type=type,[TypeOfParentMBean=NameOfParentMBean]
```

where:

- `com.bea.wlevs:` is the JMX domain name.
- `Name=`*`name`*`,Type=`*`type`*`,[`*`TypeOfParentMBean=NameOfParentMBean`*`]` is a set of JMX key properties.

The order of the key properties is not significant, but the object name must begin with `com.bea:wlevs:`.

For example, the object name of the MBean corresponding to a processor called `myprocessor` in the application `myapplication` in the domain is as follows:

```
com.bea.wlevs:Name=myprocessor,Type=CQLProcessor,Application=myapplication
```

Table 11-1 describes the key properties that Oracle Event Processing encodes in its MBean object names.

Table 11-1 Oracle Event Processing MBean Object Name Key Properties

This Key Property	Specifies
`Name=`*`name`*	The string that you provided when you created the resource that the MBean represents. This is typically the name of a component. The name of a particular component is specified in the EPN assembly file using the `id` attribute of the component registration. For example, in the case of processors, the entry in the EPN assembly file might look like the following: `<wlevs:processor id="myprocessor" advertise="true" />` In this case, the key property would be `Name=myprocessor`.
`Type=`*`type`*	The short name of the MBean's type. The short name is the unqualified type name without the `MBean` suffix. For example, for an MBean that is an instance of the `CQLProcessorMBean`, use `CQLProcessor`. In this case, the key property would be `Type=CQLProcessor`.

`TypeOfParentMBean=`*`NameOfParentMBean`*	Specifies the type and name of the parent MBean.
	For components, this is always `Application=`*`application_name`*, where *`application_name`* refers to the name of the application of which the component is a part.
	The name of a particular Oracle Event Processing application is specified with the `Bundle-SymbolicName` header of the `MANIFEST.MF` file of the application bundle. For example, if an application has the following `MANIFEST.MF` snippet (only relevant parts are shown):
	`Manifest-Version: 1.0`
	`Archiver-Version:`
	`Build-Jdk: 1.5.0_06`
	`....`
	`Bundle-SymbolicName: myapplication`
	then the key property would be `Application=myapplication`.

Table 11-2 shows examples of configuration MBean object names that correspond to the component declarations in the HelloWorld sample EPN assembly file. In each example, the application name is `helloworld` and the domain name is `myDomain`.

Table 11-2 Component Declaration Example With Corresponding MBean Object Names

EPN Assembly File Component Declaration	Corresponding Configuration MBean Object Name
`<wlevs:processor id="helloworldProcessor" />`	`com.bea.wlevs:Name=helloworldProcessor, Type=CQLProcessor,Application=helloworld,Domain=myDomain` `CQLProcessor` is the standard configuration MBean for processor components. The manageable property is `rules`.
`<wlevs:channel id="helloworldInstream"> <wlevs:listener ref="helloworldProcessor"/> <wlevs:source ref="helloworldAdapter"/> </wlevs:channel>`	`com.bea.wlevs:Name=helloworldInstream,Type=EventChannel,Application=helloworld,Domain=myDomain` `Channel` is the standard configuration MBean for a channel component. The manageable properties are `MaxSize` and `MaxThreads`.

11.3.3 Run Time MBeans

You can gather monitoring information for each component in the EPN with run time MBeans. Oracle Event Processing server defines the following metrics that you can monitor for each component:

- Throughput: The number of events processed by the component. The parameters for this metric are: throughput time interval, aggregation time interval, the unit of time for the intervals.
- Average Latency: The average amount of time it takes an event to pass through a component, or *latency*. Parameters: aggregation time interval, the unit of time for the interval.
- Maximum Latency: The maximum amount of time it takes an event to pass through a component. Parameters: aggregation time interval, the unit of time for the interval.
- Average Latency Threshold: Specifies whether the average latency of events between the start- and end-points of a component crosses a specified threshold. Parameters: aggregation time interval, threshold, the unit of time for the interval.

11.3.4 Run Time MBean Naming

Run time MBeans are named using the same pattern as configuration mbeans except for one extra property: `Direction`. This property has two valid values: `OUTBOUND` or `INBOUND` that refer to the point at which you want to gather the statistic `OUTBOUND` means that you want to gather throughput or latency as events flow out of the specified component; similarly `INBOUND` means you want to gather the monitoring information as events flow into a component.

For example, the object name of the run time MBean corresponding to a processor called `myprocessor` in the application `myapplication`, in which events will be monitored as they flow into the component, is as follows:

```
com.bea.wlevs:Name=myprocessor,Type=CQLProcessor,Application=myapplication,Direction=INBOUND
```

See Configuration MBean Naming for details about configuration MBean naming.

11.3.5 Oracle Event Processing MBean Hierarchy

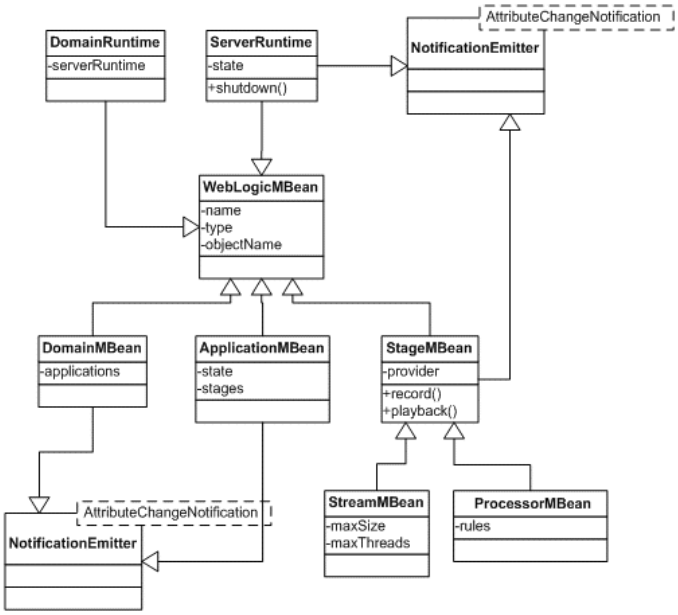
All MBeans must be registered in an MBean server under an object name of type `javax.management.ObjectName`. Oracle Event Processing follows a convention in which object names for child MBeans contain part of its parent MBean object name.

There are two main MBean roots: `DomainMBean` and `DomainRuntimeMBean`. The former includes configuration MBeans for the entire domain, the latter contains run time information, such as statistics, and local services, such as Monitor, that are generally kept to a single server instance.

`ApplicationMBean` is a child of the `DomainMBean` instead of the `ServerMBean`. This is because an application is unique within a domain, and can span multiple servers.

Figure 11-1 shows the main classes and relationships that make up the object model.

Figure 11-1 Oracle Event Processing MBean Object Model



Most MBeans emit notifications and generate `AttributeChangeNotifications`. A JMX client can register to receive attribute change notifications regarding changes to application state, insertion and removal of applications at the domain, channel size and thread changes, insertion and removal of rules, and so on.

11.4 Configure JMX

Before you can manage Oracle Event Processing applications, servers, and domains using JMX and Oracle Event Processing MBeans, you must first configure the JMX service on your Oracle Event Processing server.

11.4.1 Example JMX Configuration

The following snippet from a `config.xml` files shows a JMX configuration.

```
<config>
  <netio>
    <name>JettyNetio</name>
    <port>12345</port>
  </netio>
  <work-manager>
    <name>WM</name>
    <fairshare>5</fairshare>
    <min-threads-constraint>1</min-threads-constraint>
    <max-threads-constraint>4</max-threads-constraint>
  </work-manager>
  <jetty>
    <name>TestJetty</name>
    <work-manager-name>WM</work-manager-name>
    <network-io-name>JettyNetio</network-io-name>
  </jetty>
  <rmi>
    <name>RMI</name>
    <http-service-name>TestJetty</http-service-name>
  </rmi>
  <jndi-context>
    <name>JNDI</name>
  </jndi-context>
  <exported-jndi-context>
    <name>exportedJNDI</name>
    <rmi-service-name>RMI</rmi-service-name>
  </exported-jndi-context>
  <jmx>
    <jndi-service-name>JNDI</jndi-service-name>
    <rmi-service-name>RMI</rmi-service-name>
  </jmx>
</config>
```

11.4.2 JMX Configuration Objects

You configure the Oracle Event Processing JMX service with the following elements in the `config.xml` file that describes your Oracle Event Processing server:

- jmx
- rmi
- jndi-context
- exported-jndi-context

For information on security configuration tasks that affect JMX, see Configure JMX Security.

jmx

Table 11-3 lists the `jmx` element child elements in the `config.xml` file that you must configure.

Table 11-3 Configuration Parameters for the jmx Element

Parameter	Type	Description
`rmi-service-name`	`String`	The name of the RMI service with which the jmx server will register to receive calls.

Parameter	Type	Description
jndi-service-name	String	The name of the JNDI service to which the jmx server will bind its object.

rmi

The Oracle Event Processing RMI service provides the following:

- Ability to register a POJO interface in a server for remote method invocation from a client.
- Ability to register for any context propagation from the client to the server on a remote method invocation, intercept, and act on this propagated context in the server.

Table 11-4 lists the `rmi` element child elements in the `config.xml` file that you use to export server-side objects to remote clients.

Table 11-4 Configuration Parameters for the rmi Element

Parameter	Type	Description
heartbeat-period	int	The number of failed heartbeat attempts before triggering disconnect notifications to all registered listeners.
http-service-name	String	The name of the HTTP service used to register remote objects (such as Jetty, see Jetty.).
heartbeat-interval	int	The amount of time, in milliseconds, between heartbeats. Once the number of unsuccessful heartbeat attempts has reached the value specified by the `HeartbeatPeriod` parameter, all registered `DisconnectListener` instances are notified.
name	String	The name of this configuration object.

jndi-context

The JNDI Factory Manager is responsible for supporting JNDI in an OSGi environment. It allows JNDI providers to be supplied as OSGi bundles, and for code running inside OSGi bundles to have full access to the JNDI environment.

The Factory Manager consists of two components:

- An OSGi bundle, which provides the OSGi-specific factory management code, to look up JNDI objects using the appropriate OSGi classloader.
- JNDI *glue code*, internal to Oracle Event Processing, that initializes the JNDI environment to support the factory manager bundle.

Table 11-5 lists the `jndi-context` element child elements in the `config.xml` file that you must configure.

Table 11-5 Configuration Parameters for the jndi-context Element

Parameter	Type	Description
`default-provider`	`boolean`	If `true`, the default Oracle Event Processing JNDI provider is used. Default value is `true`.
`name`	`String`	The name of this configuration object.

exported-jndi-context

Requires a configured jndi-context.

Use this configuration object to export a remote JNDI service to a client using RMI. A JNDI context is registered with the RMI service to provide remote access to clients that pass a provider URL parameter in their `InitialContext` object.

Table 11-6 lists the `exported-jndi-context` element child elements in the `config.xml` file that you must configure.

Table 11-6 Configuration Parameters for the exported-jndi-context Element

Parameter	Type	Description
`rmi-service-name`	`String`	The name of the RMI service that should be used to serve this JNDI context over the network. It must match an existing `<rmi>` configuration object. See rmi.
`name`	`String`	The name of this configuration object. The value of this element must be different from the value of the `<name>` child element of `<jndi-context>` in the same `config.xml` file.

11.5 Manage with JMX

This section describes detailed examples of managing Oracle Event Processing components using JMX, including:

- Connect with APIs to a JMX Server from a Non-Oracle Event Processing Client
- Connect with APIs to a JMX Server From an Oracle Event Processing Client
- Configure an Oracle Event Processing Component with JMX APIs
- Monitor the Throughput and Latency of a Component with JMX APIs
- Connect to a Local or Remote JMX Server using JConsole with Security
- Connect to Local or Remote JMX Server Using JConsole with Security Disabled.

Note:

When using JConsole, you must start it with the Oracle Event Processing `wlevsjconsole.cmd` or `wlevsjconsole.sh` script. You cannot start `jconsole` directly.

11.5.1 Connect with APIs to a JMX Server from a Non-Oracle Event Processing Client

This section describes how to write Java code using the JMX API (`http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement`) to connect to the Oracle Event Processing JMX server from a non-Oracle Event Processing client. This is the first step to all programmatic JMX management.

For information on connecting to the Oracle Event Processing JMX server from another Oracle Event Processing server, see Connect with APIs to a JMX Server From an Oracle Event Processing Client.

Use APIs to connect to a JMX server from a non-Oracle Event Processing client:

1. Be sure that the JMX service is configured for your domain.

   For details see Configure JMX.

2. Write the `http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement` Java code to configure the component using the appropriate MBean.

   Consider the following JMX programming hints.

   One of the first things you must do in your JMX program is to establish a connection to the JMX server running in the Oracle Event Processing server as shown below.

```
public static void initConnection(String hostname, int port, String username, char[]
password)
  throws IOException,MalformedURLException {

  Map<String,Object> env = makeSecureEnv();
  env.put("jmx.remote.protocol.provider.pkgs","com.bea.core.jmx.remote.provider");
  env.put("mx4j.remote.resolver.pkgs","com.bea.core.jmx.remote.resolver");
  env.put("java.naming.factory.initial","com.bea.core.jndi.context.ContextFactory");

  JMXServiceURL serviceUrl = new JMXServiceURL(
      "MSARMI","localhost",9002,"/jndi/jmxconnector"
  );

  System.out.println("Service: " + serviceURL.toString());

  JMXConnector connector = JMXConnectorFactory.connect(serviceUrl,env);

  MBeanServerConnection connection = connector.getMBeanServerConnection();
}

// The JMXConnectorFactory.connect() method's second parameter is a Map object that sets up
a
// secure environment using the makeSecureEnv() method, which looks like the following:

private static Map<String,Object> makeSecureEnv() {
    Map<String,Object> env = new HashMap<String,Object>();
    String username = "wlevs" ;
    char[] password = { 'w','l','e','v','s' };
    env.put(JMXConnector.CREDENTIALS, new Serializable[]{username,password});
    env.put("jmx.remote.authenticator", "com.bea.core.jmx.server.CEAuthenticator");
    System.setProperty("jmx.remote.authenticator",
"com.bea.core.jmx.server.CEAuthenticator");
    return env;
}
```

11.5.2 Connect with APIs to a JMX Server From an Oracle Event Processing Client

This section describes how to write Java code using the JMX API (`http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement`) to connect to the Oracle Event Processing JMX server from another Oracle Event Processing server. This is the first step to all programmatic JMX management.

For information on connecting to the Oracle Event Processing JMX server from a non-Oracle Event Processing client, see Connect with APIs to a JMX Server from a Non-Oracle Event Processing Client.

Use APIs to connect to the JMX server from an Oracle Event Processing client:

1. Be sure that the JMX service is configured for your domain.

   For details see Configure JMX.

2. Write the `http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement` Java code to configure the component using the appropriate MBean.

   Consider the following JMX programming hints.

   One of the first things you must do in your JMX program is to establish a connection to the JMX server running in the Oracle Event Processing server as shown below.

```
public static void initConnection(String hostname, int port, String username, char[]
password)
  throws IOException,MalformedURLException {

  Map<String,Object> env = makeSecureEnv();

  // This is an OSGi necessity
  env.put(
      JMXConnectorFactory.DEFAULT_CLASS_LOADER,
      com.bea.core.jmx.remote.provider.msarmi.ServerProvider.class.getClassLoader()
  );
  env.put(
      JMXConnectorFactory.PROTOCOL_PROVIDER_CLASS_LOADER,
      com.bea.core.jmx.remote.provider.msarmi.ServerProvider.class.getClassLoader()
  );

  JMXServiceURL serviceUrl = new JMXServiceURL(
      "MSARMI","localhost",9002,"/jndi/jmxconnector"
  );

  System.out.println("Service: " + serviceURL.toString());

  env.put(
      JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
      "com.bea.core.jmx.remote.provider"
  );

  System.setProperty("mx4j.remote.resolver.pkgs", "com.bea.core.jmx.remote.resolver");

  JMXConnector connector = JMXConnectorFactory.connect(url, env);
  connector.connect();

  MBeanServerConnection connection = connector.getMBeanServerConnection();
  ...
}

// The JMXConnectorFactory.connect() method's second parameter is a Map object that sets up
```

```
a
// secure environment using the makeSecureEnv() method, which looks like the following:

private static Map<String,Object> makeSecureEnv() {
    Map<String,Object> env = new HashMap<String,Object>();
    String username = "wlevs" ;
    char[] password = { 'w','l','e','v','s' };
    env.put(JMXConnector.CREDENTIALS, new Serializable[]{username,password});
    env.put("jmx.remote.authenticator", "com.bea.core.jmx.server.CEAuthenticator");
    System.setProperty("jmx.remote.authenticator",
"com.bea.core.jmx.server.CEAuthenticator");
    return env;
}
```

11.5.3 Configure an Oracle Event Processing Component with JMX APIs

This section describes how to write Java code using the JMX API (`http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement`) to access Oracle Event Processing MBeans.

Configure an Oracle Event Processing component with JMX APIs:

1. Acquire a connection to the Oracle Event Processing JMX server.

   For details see Connect with APIs to a JMX Server from a Non-Oracle Event Processing Client.

2. Write the `http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement` Java code to configure the component using the appropriate MBean.

   Consider the following JMX programming hints.

   The following example shows how to use the connection to start getting information about the domain and its deployed applications by querying MBeans.

   First the code shows how to get all MBeans whose type is Domain; there should only be one. Then, using the `DomainMBean`, the sample shows how to retrieve a list of all the deployed applications in the domain (using `ApplicationMBean`):

```
Set domainObjectNames = connection.queryMBeans(
    ObjectName.getInstance(
        ManagementConstants.DOMAIN_NAME + ":" +
        ManagementConstants.TYPE_PROPERTY + "=" +
        DomainMBean.MBEAN_TYPE + ",*"
    ),
    null
);
ObjectName domainName =  ((ObjectInstance)
domainObjectNames.iterator().next()).getObjectName();
System.out.println("Domain Name: " +
domainName.getKeyProperty(ManagementConstants.NAME_PROPERTY));
ObjectName [] applicationNames =
     (ObjectName[]) connection.getAttribute(domainName, "ApplicationMBeans");
ObjectName selectedApplicationObjectName = null ;
for (ObjectName applicationName : applicationNames) {
   String name =
     applicationName.getKeyProperty(ManagementConstants.NAME_PROPERTY);
   String status =
     (String) connection.getAttribute(applicationName, "State");
    System.out.println("Application: " + name + " Status: " + status);
    selectedApplicationObjectName = applicationName ;
```

11.5.4 Monitor the Throughput and Latency of a Component with JMX APIs

This section describes how to write Java code using the JMX API (http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement) to access Oracle Event Processing MBeans and dynamically monitor the throughput and latency of an Oracle Event Processing component.

Dynamically configure an Oracle Event Processing component with JMX APIs:

1. Acquire a connection to the Oracle Event Processing JMX server.

   For details see Connect with APIs to a JMX Server from a Non-Oracle Event Processing Client.

2. Acquire an instance of a `MonitorRuntimeMBean` for the component you want to monitor as shown below.

   ```
   ObjectName processorInbound = ObjectName.getInstance(
       "com.bea.wlevs:Name=myprocessor," +
       "Type=CQLProcessor," +
       "Application=myapplication," +
       "Direction=INBOUND"
   );
   ```

   Be sure you specify whether you want to monitor incoming events (`INBOUND`) or outgoing events (`OUTBOUND`).

3. Use the `MonitorRuntimeMBean` to acquire an instance of `ProbeRuntimeMBean` for the type of statistic you want as shown below.

   ```
   ObjectName monitorName =
       ObjectName.getInstance(
           "com.bea.wlevs:ServerRuntime=localhost," +
           "Name=MonitorRuntime," +
           "Type=MonitorRuntime");

   MonitorRuntimeMBean monitorMBean =
       (MonitorRuntimeMBean)MBeanServerInvocationHandler.newProxyInstance(
           connection,
           monitorName,
           MonitorRuntimeMBean.class,
           false);

   ObjectName probeName = monitorMBean.monitorAvgThroughput(
       processorInbound,
       1000,
       1000
   );

   ProbeRuntimeMBean probeOn =
   (ProveRuntimeMBean)MBeanServerInvocationHandler.newProxyInstance(
       connection,
       probeName,
       ProbeRuntimeMBean.class,
       false
   );
   ```

   The `MonitorRuntimeMBean` has methods for each type of statistic you can gather. For example, you execute `monitorAvgLatency()` if you want to

monitor the average latency, `monitorAvgThroughput()` to monitor the average throughput, and so on. These methods all return `ProbeRuntimeMBean`.

4. Use the `ProbeRuntimeMbean` instance to get the actual run time metrics in one of the following ways:

   a. Use the `ProbeRuntimeMBean` method `getMetric()` to pull the information.

   b. Use `javax.management.NotificationBroadcaster.addNotificationListener()` to have the information pushed to you every time there is a change in the metrics.

5. When you are finished gathering monitoring information, unregister the MBean from the MBean server as shown below.

   ```
   probON.terminate();
   ```

   For additional details about these MBean interfaces and how to use them to monitor throughput and latency, see the `com.bea.wlevs.monitor.management` package in the *Java API Reference for Oracle Stream Explorer*.

11.5.5 Connect to a Local or Remote JMX Server using JConsole with Security

You can use the `wlevsjconsole` script to connect to an Oracle Event Processing JMX server running on your local host or on a remote host to browse and manage Oracle Event Processing MBeans with the JDK `jconsole`.

This procedure describes how to use JConsole when the Oracle Event Processing server has security enabled. This is the default configuration and is recommended for production servers. Alternatively, you can connect to the JMX server with security disabled (see Connect to Local or Remote JMX Server Using JConsole with Security Disabled).

For more information, see Access the Oracle Event Processing JMX Server.

Note:

When using JConsole, you must start it with the Oracle Event Processing `wlevsjconsole.cmd` or `wlevsjconsole.sh` script. You cannot start `jconsole` directly.

Connect to a local or remote JMX server using JConsole with security enabled:

1. Ensure that the local or remote Oracle Event Processing server is running.
2. Launch `jconsole` using the `wlevsjconsole.cmd` or `wlevsjconsole.sh` script located in the */Oracle/Middleware/my_oep/oep/bin* directory.

   a. To connect to a local Oracle Event Processing server, enter:

   ```
   prompt> wlevsjconsole.cmd
   ```

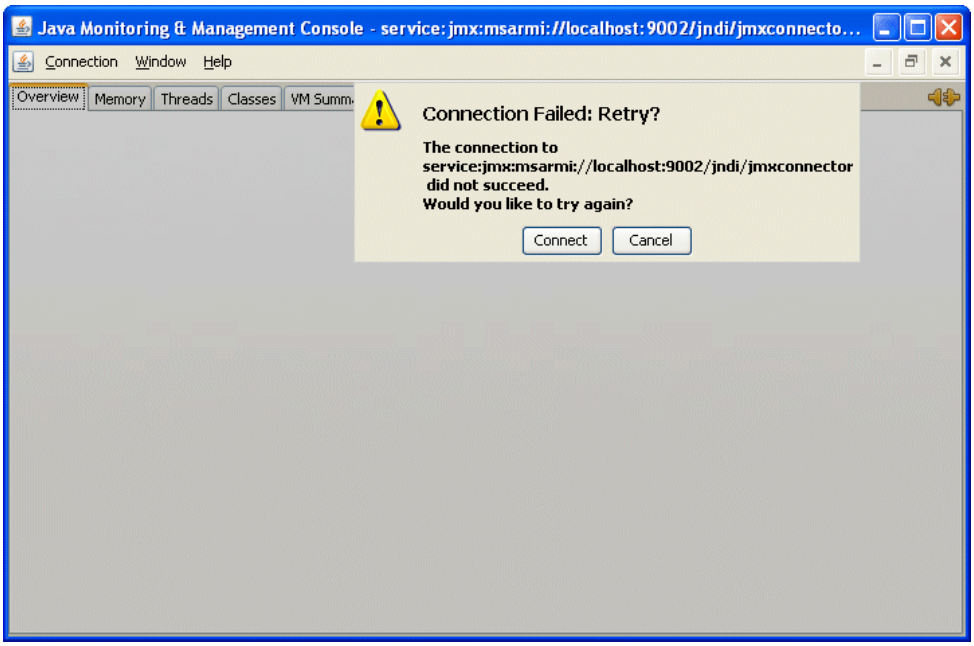
   b. To connect to a remote Oracle Event Processing server, enter:

   ```
   prompt> wlevsjconsole.cmd HOST-NAME:PORT
   ```

Where *HOST-NAME* is the name of the remote host and *PORT* is the Net IO port as configured in the remote host's */Oracle/Middleware/my_oep/user_projects/domains/myDomain/*defaultserver/config/config.xml file.

The jconsole browser attempts to log into the JMX server and initially fails as Figure 11-2 shows.

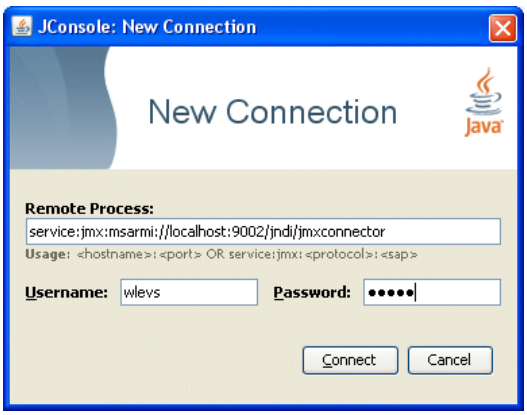
Figure 11-2 Jconsole Initial Login Attempt



3. Click **Cancel**.

   The `Jconsole` New Connection dialog appears as shown in Figure 11-3.

Figure 11-3 JConsole New Connection Dialog



4. Configure the New Connection dialog as Table 11-7 describes.

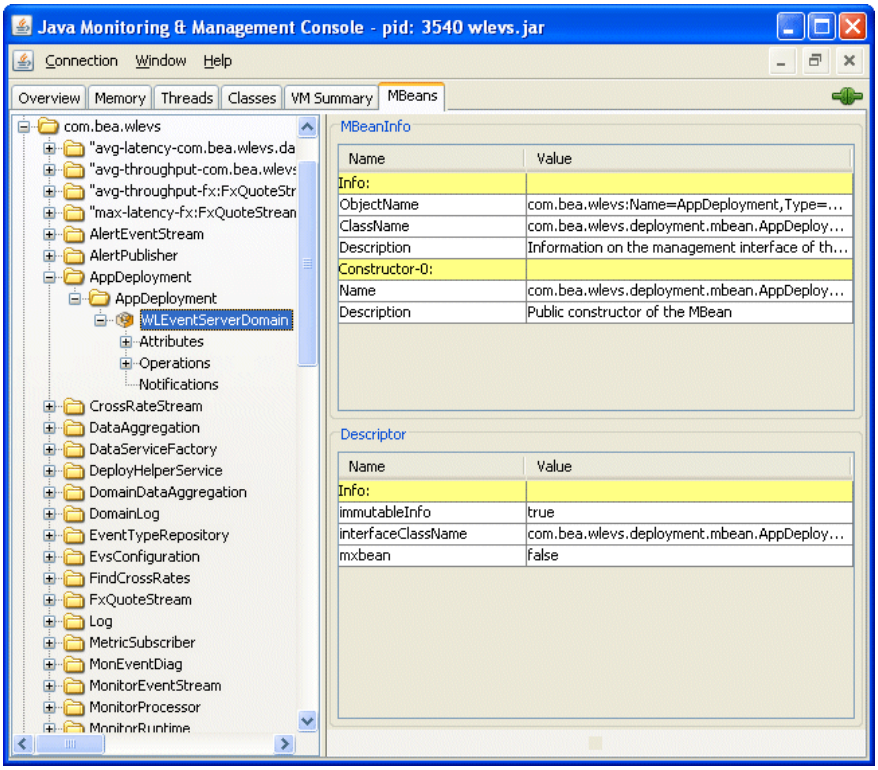
Table 11-7 JConsole New Connection Attributes

Attribute	Description
`Remote Process`	Enter the following URL: `service:jmx:msarmi://HOST-NAME:PORT/jndi/jmxconnector` Where *`HOST-NAME`* is the name of the local or remote host and *`PORT`* is the Net IO port as configured in the remote host's */Oracle/Middleware/my_oep/*`user_projects/domains/`*myDomain/defaultserver/config/config.xml* file (default: `9002`).
`Username`	Enter the Oracle Event Processing administration user name: **oepadmin**.
`Password`	Enter the Oracle Event Processing administration password: **welcome1**.

5. Click **Connect**.

   The `jconsole` browser opens and provides access to Oracle Event Processing MBeans as Figure 11-5 shows.

Figure 11-4 JConsole Browser



11.5.6 Connect to Local or Remote JMX Server Using JConsole with Security Disabled

You can use the `wlevsjconsole` script to connect to an Oracle Event Processing JMX server running on your local host or on a remote host to browse and manage Oracle Event Processing MBeans with the JDK `jconsole`.

This procedure describes how to use JConsole when the Oracle Event Processing server has security disabled. This is a common development configuration and is not recommended for production servers. Alternatively, you can connect to the JMX server with security enabled (see Connect to a Local or Remote JMX Server using JConsole with Security).

For more information, see Access the Oracle Event Processing JMX Server.

Note:

When using JConsole, you must start it with the Oracle Event Processing `wlevsjconsole.cmd` or `wlevsjconsole.sh` script. You cannot start `jconsole` directly.

Connect to a local or remote JMX server using JConsole with security disabled:

1. Ensure that the local or remote Oracle Event Processing server is running with security disabled.

   For more information, see Disable Security.

2. Launch `jconsole` using the `wlevsjconsole.cmd` or `wlevsjconsole.sh` script located in the *`/Oracle/Middleware/my_oep/oep/bin`* directory.

   a. To connect to a local Oracle Event Processing server, enter:

   ```
   prompt> wlevsjconsole.cmd
   ```

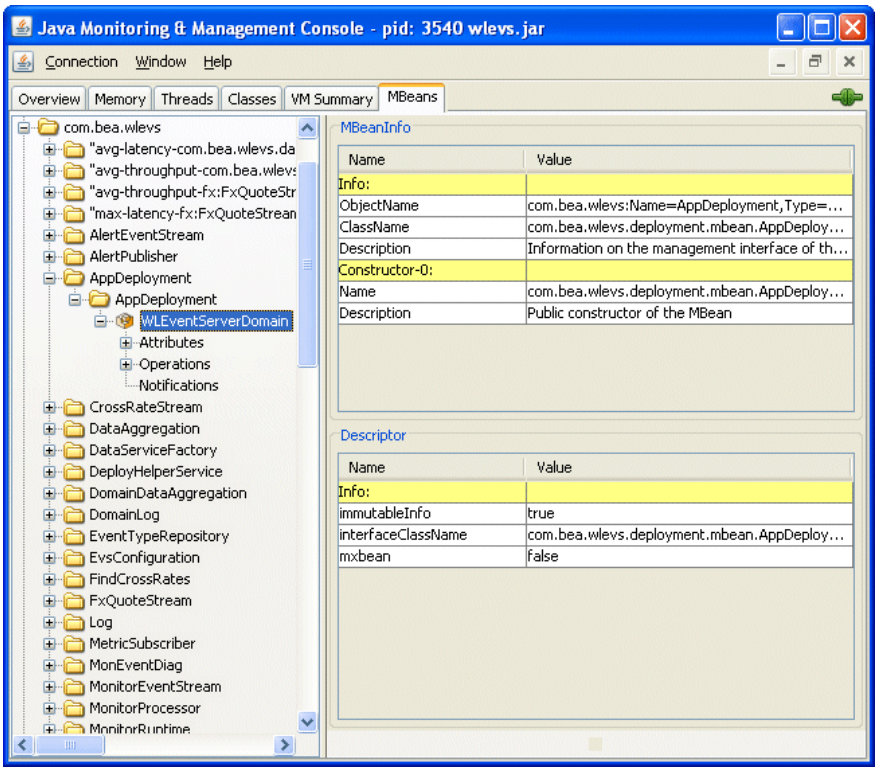
   b. To connect to a remote Oracle Event Processing server, enter:

   ```
   prompt> wlevsjconsole.cmd HOST-NAME:PORT
   ```

   Where *`HOST-NAME`* is the name of the remote host and *`PORT`* is the `Net IO` port as configured in the remote host's */Oracle/Middleware/my_oep/user_projects/domains/myDomain/defaultserver/config/config.xml* file.

   The script automatically connects to the JMX server and the `jconsole` browser opens and provides access to Oracle Event Processing MBeans as Figure 11-5 shows.

Figure 11-5 JConsole Browser



12

JDBC

You can configure Java Database Connectivity (JDBC) for relational database access from an Oracle Event Processing application.

Oracle Event Processing supports JDBC 4.0. To download JDBC, go to `http://java.sun.com/products/jdbc/download.html`.

This chapter includes the following sections:

- Database Access
- Oracle Event Processing Data Sources
- Configure Access to a Database with an Oracle JDBC Driver
- Configure Database Access with Microsoft SQL Server JDBC Driver
- Configure Access to a Different Database Driver or Driver Version.

12.1 Database Access

The JDBC API provides a standard, vendor-neutral way for applications to connect to and interact with database servers and other types of tabular resources that support the JDBC API. A database driver can implement the JDBC `javax.sql.DataSource` interface to define a database connection factory. Applications use the `DataSource` objects to obtain database connections (`java.sql.Connection`). An application obtains a connection and interacts with the data resource by sending SQL commands and receiving results.

Oracle Event Processing provides a `DataSource` abstraction that encapsulates a JDBC driver `DataSource` object and manages a pool of pre-established connections. Also, the Oracle WebLogic Server `WLConnection` interface provides methods that enable access to and manipulation of Oracle data sources. For more information, see Oracle Event Processing Data Sources.

Oracle Event Processing provides the Oracle 12c thin driver. Optionally, you can use your own JDBC driver. See Access a Database Driver with bootclasspath.

12.1.1 Oracle JDBC Driver

Oracle Event Processing includes the Oracle 12*c* Thin driver for use with Java SE 7. The JDBC Thin driver is a pure Java, Type IV driver that you can be use in applications and applets. It is platform-independent and does not require any additional Oracle software on the client side. The JDBC Thin driver communicates with the server using SQL*Net to access the Oracle Database.

The Oracle 12*c* Thin drive is in the following JAR file:

```
/Oracle/Middleware/wlserver/modules/
com.bea.oracle.ojdbc6_1.0.0.0_11-2-0-3-0.jar
```

For more information, see:

- Supported Databases
- Configure Access to a Database with an Oracle JDBC Driver
- `http://www.oracle.com/technetwork/database/application-development/index-099369.html`.

12.1.2 Supported Databases

Oracle Event Processing servers support different databases depending on the type of JDBC driver you use.

Oracle JDBC Driver

Using the Oracle JDBC driver, you can access the Oracle Database 12c release. See Oracle JDBC Driver.

SQL Server Type 4 JDBC Driver from DataDirect

Using the SQL Server Type 4 JDBC Driver from Microsoft, you can access the following SQL Server databases:

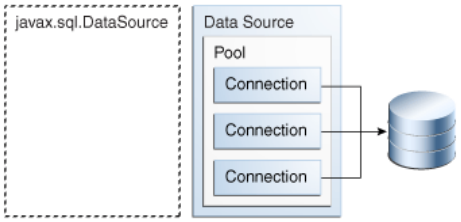
- Microsoft SQL Server 2012
- Microsoft SQL Server 2005
- Microsoft SQL Server 2000
- Microsoft SQL Server 2000 Desktop Engine (MSDE 2000)
- Microsoft SQL Server 2000 Enterprise Edition (64-bit)
- Microsoft SQL Server 7.0.

12.2 Oracle Event Processing Data Sources

An Oracle Event Processing `DataSource` provides a JDBC data source connection pooling implementation that supports the JDBC 4.0 specification. Applications reserve and release `Connection` objects from a data source with the standard `DataSource.getConnection` and `Connection.close` APIs respectively.

Figure 12-1 shows the relationship between data source, connection pool, and `Connection` instances.

Figure 12-1 Oracle Event Processing Data Source



You must use the Oracle Event Processing server default data source or configure your own Oracle Event Processing `DataSource` in the server's `config.xml` file if you want to access a relational database in any of the following ways. See *Schema Reference for Oracle Stream Explorer*.

- From an Oracle CQL processor `rule`.
- Event record and playback.
- From a cache loader or store.

You do not have to configure a `DataSource` in the server's `config.xml` file if you use the JDBC driver's API, such as `DriverManager`, directly in your application code.

12.2.1 Default Data Source Configuration

By default, the Oracle Event Processing server creates a local transaction manager. The transaction manager depends on a configured RMI object, as described in JMX Configuration Objects. Oracle Event Processing server guarantees that there will never be more than one transaction manager instance in the system.

If a database is unavailable when you start Oracle Event Processing server, by default, an Oracle Event Processing server data source retries every 10 seconds until it creates a connection. The retries enable the Oracle Event Processing server to start when a database is unavailable. You can change the retry interval by providing a value for the `connection-creation-retry-frequency-seconds` child element of the `connection-pool-params` element. A value of zero disables connection retry.

12.2.2 Custom Data Source Configuration

The Oracle Event Processing server `config.xml` file requires a configuration element for each data source that is to be created at runtime that references an external JDBC module descriptor.

When you create an Oracle Event Processing domain with the Configuration Wizard, you can optionally configure a JDBC data source that uses one of the two supported Data Direct JDBC drivers. In this case the wizard updates the `config.xml` file for you. When you configure the data source, you provide basic information, such as the database you want to connect to, and the connection user name and password.

You can also update the `config.xml` file manually by adding a `data-source` element as the following example shows.

```
<data-source>
    <name>rdbms</name>
    <data-source-params>
        <global-transactions-protocol>None</global-transactions-protocol>
    </data-source-params>
    <connection-pool-params>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <initial-capacity>5</initial-capacity>
        <max-capacity>10</max-capacity>
        <connection-creation-retry-frequency-seconds>
            60
        </connection-creation-retry-frequency-seconds>
    </connection-pool-params>
    <driver-params>
        <url>jdbc:oracle:thin:@localhost:5521:rdb</url>
        <driver-name>oracle.jdbc.OracleDriver</driver-name>
        <properties>
            <element><name>user</name><value>scott</value></element>
            <element><name>password</name><value>tiger</value></element>
        </properties>
        <use-xa-data-source-interface>true</use-xa-data-source-interface>
    </driver-params>
</data-source>
```

```
<transaction-manager>
    <name>TM</name>
    <rmi-service-name>RMI</rmi-service-name>
</transaction-manager>
```

A data source depends on the availability of a local transaction manager. You can rely on the default Oracle Event Processing server transaction manager or configure one with the `transaction-manager` element in `config.xml` as the example shows. The transaction manager depends on a configured RMI object, as described in JMX Configuration Objects.

If a database is unavailable when you start Oracle Event Processing server, by default, an Oracle Event Processing server data source retries every 10 seconds until creates a connection. The retries enable Oracle Event Processing server to start when a database is unavailable. The example changes the retry interval in the `config.xml` file by providing a value for the `connection-creation-retry-frequency-seconds` child element of the `connection-pool-params` element. In the example, the value is 60 seconds.

For a full list of child elements of the `data-source` element such as the `connection-pool-params` and `data-source-params` elements, see *Schema Reference for Oracle Stream Explorer*. For information about security configuration tasks that affect JDBC, see Configure JDBC Security.

12.2.3 Get the Native JDBC Connection

The *Java API Reference for Oracle WebLogic Server* provides a `WLConnection` interface that contains methods for getting and manipulating Oracle data sources. For example, the following Java code gets the native Oracle database connection from the pooled connection object.

```
private DataSource ods;
private Connection wlConnection;
private OracleConnection connection;

wlConnection = ods.getConnection();
connection = (OracleConnection) ((WLConnection) wlConnection)
.getVendorConnection();
```

Note:

Close pooled connections when you finish and do not use a native connection object after the pooled connection is closed.

12.3 Configure Access to a Database with an Oracle JDBC Driver

This section explains the procedure to configure access to a database with an Oracle JDBC driver.

The Oracle JDBC driver is installed with Oracle Event Processing and ready to use.

Configure access to a database using the Oracle JDBC driver:

1. Configure the data source in the server `config.xml` file.
   a. To update the Oracle Event Processing server `config.xml` file using the Configuration Wizard, see Create a Standalone-Server Domain.

b. To update the Oracle Event Processing server `config.xml` file manually, see Custom Data Source Configuration.

   The `url` element for the Oracle JDBC driver has the following form. See also Custom Data Source Configuration.

   ```
   <url>jdbc:oracle:thin:@HOST:PORT:SID</url>
   ```

2. If Oracle Event Processing is running, restart it so it reads the new data source information.

   For more information, see "Start and Stop Servers".

12.4 Configure Database Access with Microsoft SQL Server JDBC Driver

To access a data source with a Microsoft SQL server JDBC driver, add the `wlsqlserv.jar` and the `fmwgenerictoken.jar` files to the `-Xbootclasspath` as follows:

```
-Xbootclasspath/a:/Oracle/Middleware/my_oep/oracle_common/modules/datadirect/
wlsqlserver.jar:/Oracle/Middleware/my_oep/oracle_common/modules/datadirect/
fmwgenerictoken.jar

-Xbootclasspath/a:/Oracle/Middleware/oracle_common/modules/datadirect/
wlsqlserver.jar:/Oracle/Middleware/oracle_common/modules/datadirect/
fmwgenerictoken.jar
```

Add the following SQL server data source configuration to the `config.xml` file.

```
<data-source>
  <name>ds-sqlserver-datadirect-driver</name>
  <data-source-params>
    <jndi-names />
    <global-transactions-protocol>OnePhaseCommit
    </global-transactions-protocol>
  </data-source-params>
  <connection-pool-params>
   <credential-mapping-enabled></credential-mapping-enabled>
   <test-table-name>SQL SELECT 1</test-table-name>
   <initial-capacity>5</initial-capacity>
   <max-capacity>20</max-capacity>
   <capacity-increment>1</capacity-increment>
  </connection-pool-params>
  <driver-params>
    <use-xa-data-source-interface>true</use-xa-data-source-interface>
    <driver-name>weblogic.jdbc.sqlserver.SQLServerDriver</driver-name>
    <url>
 jdbc:weblogic:sqlserver://hostname:port;databaseName=fmwcerts;SelectMethod=cursor
    </url>
    <properties>
      <element>
        <value>sa</value>
        <name>user</name>
      </element>
      <element>
        <value>{AES}XcrEKM8RegvOT3jZ4d46WQ==</value>
        <name>password</name>
      </element>
    </properties>
  </driver-params>
</data-source>
```

12.5 Configure Access to a Different Database Driver or Driver Version

In some cases, you might need to use a different version of the Oracle Database driver or Data Direct drivers than the version bundled with Oracle Event Processing, or you might need to use a database driver other than the Oracle Database driver or Data Direct drivers.

12.5.1 Access a Database Driver with an Application Library Built With bundler.sh

This procedure describes how to create an OSGi bundle for your driver using the bundler utility and deploy it on the Oracle Event Processing server.

See *Developing Applications for Event Processing with Oracle Stream Explorer*.

1. Execute the `bundler.sh` script to create an OSGi bundle that contains your driver.

   The `bundler.sh` script is in the */Oracle/Middleware/my_oep/oep/bin* directory. The following example lists the `bundler.sh` command-line options and Table 12-1 describes them.

   Note:

   There is no Windows support for bundler.sh (no bundler.cmd).

```
bundler.sh
    -source <jar>
    -name <name>
    -version <version>
    [-factory <class>+]
    [-service <interface>+]
    [-stagedir <path>]
    [-targetdir <path>]
```

Table 12-1 bundler.sh Command Line Options

Argument	Description
`-source`	The path of the source JAR file to be bundled.
`-name`	The symbolic name of the bundle. The root of the target JAR file name is derived from the name value.
`-version`	The bundle version number. All exported packages are qualified with a version attribute with this value. The target JAR file name contains the version number.
`-factory`	An optional argument that specifies a space-delimited list of one or more factory classes that are to be instantiated and registered as OSGi services. Each service is registered with the OSGi service registry with name (`-name`) and version (`-version`) properties.
`-service`	An optional argument that specifies a space-delimited list of one or more Java interfaces that are used as the object class of each factory object service registration. If no interface names are specified, or the number of interfaces specified does not match the number of factory classes, then each factory object will be registered under the factory class name.

Argument	Description
`-stagedir`	An optional argument that specifies where to write temporary files when creating the target JAR file. Default: `./bundler.tmp`
`-targetdir`	An optional argument that specifies the location of the generated bundle JAR file. Default: current working directory (`.`).

The following example shows how to use the `bundler.sh` to create an OSGi bundle for an Oracle JDBC driver.

```
bundler.sh \
    -source /scratch/drivers/com.bea.oracle.ojdbc6_1.0.0.0_11-2-0-3-0.jar \
    -name oracle12c \
    -version 12.1.3 \
    -factory oracle.jdbc.xa.client.OracleXADataSource oracle.jdbc.OracleDriver \
    -service javax.sql.XADataSource java.sql.Driver \
    -targetdir /scratch/stage
```

The source JAR is an Oracle driver located in the `C:\drivers` directory. The name of the generated bundle JAR is the concatenation of the `-name` and `-version` arguments and is created in the `C:\stage` directory. The bundle JAR contains the files that the following example shows:

```
1465 Thu Jun 29 17:54:04 EDT 2006 META-INF/MANIFEST.MF
1540457 Thu May 11 00:37:46 EDT 2006 ccom.bea.oracle.ojdbc6_1.0.0.0_
  11-2-0-3-0.jar
1700 Thu Jun 29 17:54:04 EDT 2006 com/bea/core/tools/bundler/Activator.class
```

The command-line options specify that there are two factory classes to instantiate and register as an OSGi service when the bundle is activated, each under a separate object class as the following table shows.

Table 12-2 Factory Class and Service Interface

Factory Class	Service Interface
`oracle.jdbc.xa.client.OracleXADataSource`	`javax.sql.XADataSource`
`oracle.jdbc.OracleDriver`	`java.sql.Driver`

Each service registration will be made with a `name` property set to `oracle12c` and a `version` property with a value of `12c`. the following example shows the Oracle Event Processing server log messages showing the registration of the services.

```
...
INFO: [Jun 29, 2006 5:54:18 PM] Service REGISTERED: { version=12c,
name=oracle12c, objectClass=[ javax.sql.XADataSource ], service.id=23 }
INFO: [Jun 29, 2006 5:54:18 PM] Service REGISTERED: { version=12c,
name=oracle12c, objectClass=[ java.sql.Driver ], service.id=24 }
INFO: [Jun 29, 2006 5:54:18 PM] Bundle oracle12c STARTED
...
```

2. Copy the bundler JAR to the Oracle Event Processing server library extensions directory.

   Because your Oracle Event Processing application is an application library that contains a driver, you copy it to the Oracle Event Processing server library

extensions directory. By default the library extensions directory is in */Oracle/Middleware/user_projects/domains/<domainname>/ <servername>/modules/ext/*.

3. In the Oracle Event Processing server `config.xml` file, create a custom `data-source` element for your driver version and add a `driver-params` child element as the following example shows.

   For more information, see Server Configuration Files.

```
<driver-params>
  <url>jdbc:oracle:thin:@lcw2k18:1531:lcw101</url>
  <driver-name>oracle.jdbc.xa.client.OracleXADataSource</driver-name
  <properties>
    <element>
      <name>user</name>
      <value>scott</value>
    </element>
    <element>
      <name>password</name>
      <value>{3DES}EoIfSBMhnW8=</value>
    </element>
    <element>
      <name>com.bea.core.datasource.serviceName</name>
      <value>oracle12c</value>
    </element>
    <element>
      <name>com.bea.core.datasource.serviceVersion</name>
      <value>12.1.3</value>
    </element>
    <element>
      <name>com.bea.core.datasource.serviceObjectClass</name>
      <value>javax.sql.XADataSource</value>
    </element>
  </properties>
  <use-xa-data-source-interface>true</use-xa-data-source-interface>
</driver-params>
```

   Table 12-1 describes the relevant properties.

Table 12-3 driver-params Properties

Property	Description
`com.bea.core.datasource.serviceName`	Specifies the value of the `serviceName` registration property. This must match the `NAME` property in your `Activator` class.
`com.bea.core.datasource.serviceVersion`	Specifies the value of the `serviceVersion` registration property. This must match the `VERSION` property in your `Activator` class.
`com.bea.core.datasource.serviceObjectClass`	Specifies the interface name of the OSGI service registration.

4. Stop and start the Oracle Event Processing server.

   For more information, see Start and Stop Servers.

12.5.2 Access a Database Driver with bootclasspath

Optionally, you can use the bootclasspath to access your own JDBC driver.

Oracle recommends that you use an application library instead, as described in Access a Database Driver with an Application Library Built With bundler.sh.

1. Go to the server directory of the domain that you want to configure.

   By default, the server directory is in `/Oracle/Middleware/my_oep/user_projects/domains/<domainname/<servername>/`.

2. In a text editor, open the start script for your platform.

3. Add the `-Xbootclasspath/a` option to the Java command that executes the `wlevs.jar` file and set the `-Xbootclasspath/a` option to the full pathname of the JDBC driver you are going to use.

   For example, to use the Windows Oracle thin driver, update the `java` command in the start script as follows with everything on one line:

   ```
   %JAVA_HOME%\bin\java -Dwlevs.home=%USER_INSTALL_DIR% -Dbea.home=%BEA_HOME%
   -Xbootclasspath/a:\Oracle\Middleware\my_oep\oep\bin
   \com.bea.oracle.ojdbc14_10.2.0.jar
   -jar "%USER_INSTALL_DIR%\bin\wlevs_3.0.jar" -disablesecurity %1 %2 %3 %4 %5 %6
   ```

4. Configure the data source in the server's `config.xml` file:

   a. To update the Oracle Event Processing server `config.xml` file using the Configuration Wizard, see Create a Standalone-Server Domain.

   b. To update the Oracle Event Processing server `config.xml` file manually, see Custom Data Source Configuration.

5. If Oracle Event Processing is running, restart it so it reads the new `java` option and data source information.

   For more information, see "Start and Stop Servers".

13

HTTP Publish-Subscribe Server

An HTTP Publish-Subscribe (HTTP pub-sub) server enables web clients to subscribe to channels and publish messages to the channels using asynchronous messaging over HTTP. You can configure an HTTP pub-sub server to subscribe to channels, use a specific type of transport, set a time out, indicate a specific work manager to deliver messages to clients, and so on.

This chapter includes the following sections:

13.1 Default HTTP Pub-Sub Server

Every Oracle Event Processing server has a default HTTP pub-sub server. Oracle Event Processing Visualizer and the Record and Playback sample application use the default HTTP pub-sub server internally. An installation can use the default HTTP pub-sub server, or have a software developer create a custom HTTP pub-sub server.

In Oracle Event Processing, HTTP pub-sub server instances are configured in the `config.xml` file of the server instance. System administrator uses the `config.xml` to configure the name of the HTTP pub-sub server, specify the transport and other parameters. An administrator then uses Oracle Event Processing Visualizer to add new channels and configure security for the channels. See *Using Visualizer for Oracle Stream Explorer*.

The <`http-pubsub`> element in the `config.xml` file contains the HTTP pub-sub server configuration. By default, the HTTP pub-sub server has the following configuration. See *Schema Reference for Oracle Stream Explorer* for information about the supported configuration settings.

```
<http-pubsub>
  <name>pubsub</name>
  <path>/pubsub</path>
  <pub-sub-bean>
    <server-config>
      <name>/pubsub</name>
      <supported-transport>
        <types>
          <element>long-polling</element>
        </types>
      </supported-transport>
      <publish-without-connect-allowed>true</publish-without-connect-allowed>
```

```
 </server-config>
   <channels>
     <element>
       <channel-pattern>/evsmonitor</channel-pattern>
     </element>
      <element>
        <channel-pattern>/evsalert</channel-pattern>
      </element>
      <element>
        <channel-pattern>/evsdomainchange</channel-pattern>
      </element>
    </channels>
  </pub-sub-bean>
</http-pubsub>
```

- <path>/pubsub</path>: The URL to the HTTP pub-sub server. In a browser, you can locate the HTTP pub-sub server by typing `http://host:port/pubsub`, where *`host`* and *`port`* refer to the computer on which Oracle Event Processing is running and the port number to which it listens, respectively. For example, `http://myhost.com:9102/pubsub`.
- <supported-transport>: The transport mode. The default is `long-polling` transport.
- <publish-without-connect-allowed>true</publish-without-connect-allowed>: Allows clients to publish messages to a channel without having to explicitly connect to the HTTP pub-sub server.
- Includes the following three channels that are used internally by Oracle Event Processing Visualizer. Do not delete these channels:
  - `/evsmonitor`
  - `/evsalert`
  - `/evsdomainchange`

13.2 HTTP Publish-Subscribe Adapters

Oracle Event Processing provides two built-in adapters that provide HTTP pub-sub server functionality in applications. A software developer adds these adapters to an application to publish messages or subscribe to a server to receive messages.

In an application, a software developer uses the HTTP pub-sub adapters as follows:

- Uses the HTTP Publisher adapter to send JavaScript Object Notation (JSON) event data out of the EPN to a web-based user interface.
- Uses the HTTP Subscriber adapter to accept JavaScript Object Notation (JSON) event data entering the EPN. JSON event data comes from an HTTP server where user actions generate events.

If a software developer needs the HTTP pub-sub server to perform additional steps such as monitoring, collecting, or interpreting incoming messages from clients, then he or she must use the server-side HTTP pub-sub server APIs to program this functionality. See the following packages:

- `com.bea.wlevs.adapters.httppubsub.api`
- `com.bea.wlevs.adapters.httppubsub.support`

See *Java API Reference for Oracle Stream Explorer*.

13.3 Server Architecture

The HTTP Publish-Subscribe server enables clients to subscribe to a channel (similar to a topic in JMS) and receive messages as they become available. In contrast traditional web applications require that all communication be initiated by the client. The server can only push updated data to its clients if it receives an explicit request. However, dynamic real-time applications require that the server send data regardless of whether the client explicitly requested it.

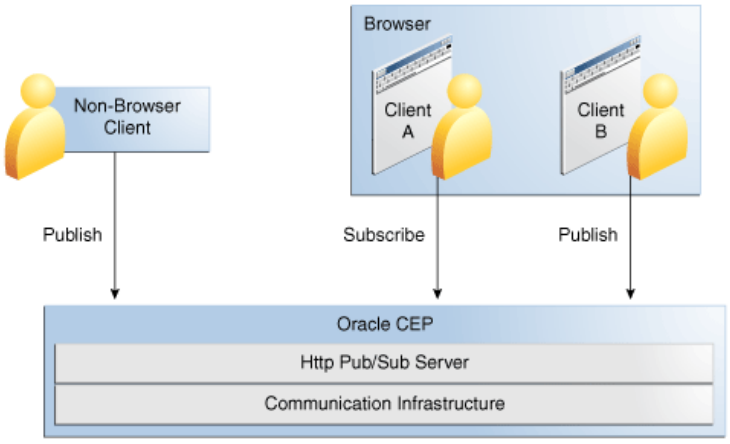
The HTTP pub-sub server is based on the Bayeux protocol proposed by the cometd project. The Bayeux protocol defines a contract between the client and the server for communicating with asynchronous messages over HTTP. It allows clients to register and subscribe to channels that are named destinations or sources of events. Registered clients, or the HTTP pub-sub server itself, then publishes messages to these channels which in turn any subscribed clients receive.

The HTTP pub-sub server can communicate with any client that can understand the Bayeux protocol. The HTTP pub-sub server is responsible for identifying clients, negotiating trust, exchanging Bayeux messages, and pushing event messages to subscribed clients. For Web 2.0 Ajax clients (such as Dojo) or rich internet applications (such as Adobe Flex) to communicate with the HTTP pub-sub server, the clients need a library that supports the Bayeux protocol. The Dojo JavaScript library provides four different transports, of which two are supported by the HTTP pub-sub server: long-polling and callback-polling.

There is a one-to-one relationship between a servlet and an HTTP pub-sub server. Each servlet has access to one unique HTTP pub-sub server. Each HTTP pub-sub server has its own list of channels. The servlet uses a context object to obtain a handle to its associated HTTP pub-sub server.

Figure 13-1 shows the basic Oracle Event Processing HTTP pub-sub server architecture.

Figure 13-1 HTTP Publish-Subscribe Server in Oracle Event Processing



13.4 Create a New HTTP Publish-Subscribe Server

The following procedure describes how to create a new HTTP pub-sub server. See Default HTTP Pub-Sub Server for a full example from the `config.xml` of a configured HTTP pub-sub server.

Create a New HTTP Publish-Subscribe Server

1. If the Oracle Event Processing server is running, stop it.

   See Start and Stop Servers.

2. In an XML editor, open the Oracle Event Processing server `config.xml` file.

   By default this file is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config`.

3. Add an `http-pubsub` child element of the root `config` element of `config.xml`, with `name`, `path` and `pub-sub-bean` child elements, as shown in bold:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:config xmlns:ns2="http://www.bea.com/ns/wlevs/config/server">
    <domain>
        <name>myDomain</name>
    </domain>
    ....
    <http-pubsub>
      <name>myPubSubServer</name>
      <path>/myPath</path>
      <pub-sub-bean>
         ...
      </pub-sub-bean>
    </http-pubsub>
    ...
</ns2:config>
```

   a. Set the `name` element to the internal name of the HTTP pub-sub server.

   b. Set the `path` element to the string that you want to appear in the URL for connecting to the HTTP pub-sub server.

   c. The next step describes the `pub-sub-bean` element.

4. Add `server-config` and `channels` child elements of the `pub-sub-bean` element:

```
<http-pubsub>
  <name>myPubSubServer</name>
  <path>/myPath</path>
  <pub-sub-bean>
    <server-config>
       ...
    </server-config>
    <channels>
       ...
    </channels>
  </pub-sub-bean>
</http-pubsub>
```

5. Update the `server-config` child element of the `pub-sub-bean` element with HTTP pub-sub server configuration as required.

   For the full list of possible elements, see *Schema Reference for Oracle Stream Explorer*.

   The following are the most common configuration options:

   - Add a `supported-transport` element to specify the transport.

The format of this element is as follows:

```
<server-config>
    <supported-transport>
        <types>
            <element>long-polling</element>
        </types>
    </supported-transport>
...
</server-config>
```

Oracle Event Processing server supports the following transports:

– `long-polling`: Using this transport, the client requests information from Oracle Event Processing server and if Oracle Event Processing server does not have information available, it does not reply until it has. When the Oracle Event Processing server replies, the client typically sends another request immediately.

– `callback-polling`: Use this transport for HTTP publish-subscribe applications using a cross domain configuration in which the browser downloads the page from one Web server (including the JavaScript code) and connects to another server as an HTTP publish-subscribe client. This is required by the Bayeux protocol.

- Add a `publish-without-connect-allowed` element to specify whether clients can publish messages without having explicitly connected to the HTTP pub-sub server; valid values are `true` or `false`:

```
<server-config>
...
    <publish-without-connect-allowed>true</publish-without-connect-allowed>
</server-config>
```

- Add a `work-manager` element to specify the name of the work manager that delivers messages to clients. The value of this element corresponds to the value of the `<name>` child element of the `work-manager` you want to assign.

```
<server-config>
...
    <work-manager>myWorkManager</work-manager>
</server-config>
```

See *Schema Reference for Oracle Stream Explorer*.

- Add a `client-timeout-secs` element to specify the number of seconds after which the HTTP pub-sub server disconnects a client if the client has not sent back a connect/reconnect message.

```
<server-config>
...
    <client-timeout-secs>600</client-timeout-secs>
</server-config>
```

6. Update the `channels` child element with at least one channel pattern.

Channel patterns always begin with a forward slash (`/`). Clients subscribe to these channels to either publish or receive messages. Add a channel pattern as shown:

```
<channels>
    <element>
```

```
            <channel-pattern>/mychannel</channel-pattern>
        </element>
    </channels>
```

7. Save the `config.xml` file.
8. Start the Oracle Event Processing server.

   See Start and Stop Servers.
9. Use Oracle Event Processing Visualizer to configure or add channels. See:
   - See *Using Visualizer for Oracle Stream Explorer*
10. Use Oracle Event Processing Visualizer to configure security for the channels. See:
    - See *Using Visualizer for Oracle Stream Explorer*
    - Configure HTTP Publish-Subscribe Server Channel Security.

13.5 Configure an Existing HTTP Publish-Subscribe Server

The following procedure describes how to configure an existing HTTP pub-sub server. See Default HTTP Pub-Sub Server for a full example from the `config.xml` of a configured HTTP pub-sub server.

Configure an Existing HTTP Publish-Subscribe Server

1. If the Oracle Event Processing server is running, stop it.

   See Start and Stop Servers.
2. In an XML editor, open the Oracle Event Processing server `config.xml` file.

   By default this file is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>/config`.
3. Search for the `http-pubsub` element that corresponds to the HTTP pub-sub server you want to configure.

   For example, to configure the default HTTP pub-sub server, locate the following entry:

   ```
   <http-pubsub>
       <name>pubsub</name>
       <path>/pubsub</path>
       <pub-sub-bean>
           <server-config>
           ...
   </http-pubsub>
   ```
4. Update the `server-config` child element of the `pub-sub-bean` element, which is a child element of `http-pubsub`, with HTTP pub-sub server configuration as required.

   For the full list of possible elements, see *Schema Reference for Oracle Stream Explorer*.

   The following are the most common configuration options:

   - Add a `supported-transport` element to specify the transport.

The format of this element is as follows:

```
<server-config>
    <supported-transport>
        <types>
            <element>long-polling</element>
        </types>
    </supported-transport>
...
</server-config>
```

Oracle Event Processing server supports the following transports:

– `long-polling`: Using this transport, the client requests information from Oracle Event Processing server and if Oracle Event Processing server does not have information available, it does not reply until it has. When the Oracle Event Processing server replies, the client typically sends another request immediately.

– `callback-polling`: Use this transport for HTTP publish-subscribe applications using a cross domain configuration in which the browser downloads the page from one web server (including the JavaScript code) and connects to another server as an HTTP publish-subscribe client. The Bayeux protocol requires this. For more about the Bayeux protocol, see `http://svn.cometd.org/trunk/bayeux/bayeux.html`.

- Add a `publish-without-connect-allowed` element to specify whether clients can publish messages without having explicitly connected to the HTTP pub-sub server; valid values are `true` or `false`:

```
<server-config>
...
    <publish-without-connect-allowed>true</publish-without-connect-allowed>
</server-config>
```

- Add a `work-manager` element to specify the name of the work manager that delivers messages to clients. The value of this element corresponds to the value of the `name` child element of the `work-manager` you want to assign.

```
<server-config>
...
    <work-manager>myWorkManager</work-manager>
</server-config>
```

See *Using Visualizer for Oracle Stream Explorer*.

- Add a `client-timeout-secs` element to specify the number of seconds after which the HTTP pub-sub server disconnects a client if the client does has not sent back a connect/reconnect message.

```
<server-config>
...
    <client-timeout-secs>600</client-timeout-secs>
</server-config>
```

5. Save the `config.xml` file.

6. Start the Oracle Event Processing server.

   See Start and Stop Servers.

7. Use Oracle Event Processing Visualizer to configure or add channels. See:
   - See *Using Visualizer for Oracle Stream Explorer*.
8. Use Oracle Event Processing Visualizer to configure security for the channels. See:
   - See *Using Visualizer for Oracle Stream Explorer*.
   - Configure HTTP Publish-Subscribe Server Channel Security.

14

Logging and Debugging

System administrators and developers configure logging output and filter log messages to troubleshoot errors or to receive notification for specific events.

Oracle Event Processing supports the Commons Apache Logging Framework, OSGi Framework Logger, and Log4j Logger logging systems. Oracle Event Processing also provides a variety of debugging options that you can enable and disable to help diagnose your Oracle Event Processing applications.

This chapter includes the following sections:

- Logging Configuration Scenarios
- Commons Apache Logging Framework
- OSGi Framework Logger
- Log4j Logger
- Configure the Logging Service
- Configure Log4j Logging
- Use the Apache Commons Logging API
- Configure Debugging Options.

For information about Oracle Event Processing security auditor logging, see Security Auditor. For information about how to parse message catalogs to validate and generate classes used for localizing text in log messages, see *Developing Applications for Event Processing with Oracle Stream Explorer*.

14.1 Logging Configuration Scenarios

The following tasks describe some logging configuration scenarios:

- Stop `DEBUG` and `INFO` messages from going to the log file.
- Allow `INFO` level messages from the HTTP subsystem to be published to the log file, but not to standard out.
- Specify that a handler publishes messages that are `WARNING` level or higher.
- Specify a default logging level for the entire server, and have a specific module override the default logging level. For example, the default logging level of the server could be `WARNING` while the logging level of the module is `DEBUG`.
- Configure a logging level for a deployed application. The application must use the Commons Apache Logging Framework if it is required to output log messages to the single server-wide log file to which server modules also log their messages.

14.2 Commons Apache Logging Framework

This section explains the commons Apache logging framework.

Oracle Event Processing provides a `commons-logging` interface. The interface provides `commons.logging.LogFactory` and `Log` interface implementations. It includes an extension of the `org.apache.commons.logging.LogFactory` class that acts as a factory to create an implementation of the `org.apache.commons.logging.Log` that delegates to the `LoggingService` in the logging module. The name of this default implementation is `weblogic.logging.commons.LogFactoryImpl`. See http://jakarta.apache.org/commons/logging/apidocs/index.html.

14.2.1 Set the Log Factory

The following list provides information about setting the log factory using system properties:

- The highest priority is given to the system property `org.apache.commons.logging.LogFactory`.
- You can set logging from the command line using:

  ```
  -Dorg.apache.commons.logging.LogFactory=weblogic.logging.commons.LogFactoryImpl
  ```

- You can programmatically implement the logging by:

  ```
  import org.apache.commons.logging.LogFactory;
  System.setProperty(
      LogFactory.FACTORY_PROPERTY,
      "weblogic.logging.commons.LogFactoryImpl"
  );
  ```

- The `weblogic.logging.commons.LogFactoryImpl` is the default log factory, if not explicitly set.
- To use another logging implementation, you must use the standard commons logging factory implementation. The `org.apache.commons.logging.impl.LogFactoryImpl` implementation is available in the commons logging jar. For example:

  ```
  -
  Dorg.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryI
  mpl
  ```

  or the equivalent programming would be:

  ```
  System.setProperty(
      LogFactory.FACTORY_PROPERTY,
      "org.apache.commons.logging.impl.LogFactoryImpl"
  );
  ```

14.2.2 Use Log Severity Levels

Each log message has an associated severity level. The level gives a rough guide to the importance and urgency of a log message. Predefined severities, ranging from `TRACE` to `EMERGENCY`, are converted to a log level when dispatching a log request to the logger. A log level object can specify any of the following values, from lowest (left-most) to highest (right-most) impact:

`TRACE`, `DEBUG`, `INFO`, `NOTICE`, `WARNING`, `ERROR`, `CRITICAL`, `ALERT`, `EMERGENCY`

You can set a log severity level on the logger, the handler, and a user application. When set on the logger, none of the handlers receive an event which is rejected by the logger. For example, if you set the log level to `NOTICE` on the logger, none of the handlers will receive `INFO` level events. When you set a log level on the handler, the restriction only applies to that handler and not the others. For example, turning `DEBUG` off for the File Handler means no `DEBUG` messages will be written to the log file, however, `DEBUG` messages will be written to standard out.

Users (Oracle Event Processing module owners or owners of user applications) are free to define the names that represent the logging category type used by the Apache commons logging for individual modules. However if the category names are defined as package names then based on the naming convention, a logging level hierarchy is assumed by default. For example, if two modules name their logging category names `com.oracle.foo` and `com.oracle.foo.bar`, then `com.oracle.foo` becomes the root node of `com.oracle.foo.bar`. This way any logging level applied to parent node (`com.oracle.foo`) automatically applies to `com.oracle.foo.bar`, unless the child node overrides the parent.

In other words, if the logging severity is specified for a node, it is effective unless the severity is inherited from the nearest parent whose severity is explicitly configured. The root node is always explicitly configured, so if nothing else if set, then all the nodes inherit the severity from the root.

Table 14-1 lists the severity levels of log messages.

Table 14-1 Log Message Severity

Severity	Meaning
`TRACE`	Used for messages from the Diagnostic Action Library. Upon enabling diagnostic instrumentation of server and application classes, `TRACE` messages follow the request path of a method.
`DEBUG`	A debug message was generated.
`INFO`	Used for reporting normal operations, a low-level informational message.
`NOTICE`	An informational message with a higher level of importance.
`WARNING`	A suspicious operation or configuration has occurred but it might not affect normal operation.
`ERROR`	A user error has occurred. The system or application can handle the error with no interruption and limited degradation of service.
`CRITICAL`	A system or service error has occurred. The system can recover but there might be a momentary loss or permanent degradation of service.
`ALERT`	A particular service is in an unusable state while other parts of the system continue to function. Automatic recovery is not possible; the immediate attention of the administrator is needed to resolve the problem.

Severity	Meaning
EMERGENCY	The server is in an unusable state. This severity indicates a severe system failure or panic.

The system generates many messages of lower severity and fewer messages of higher severity. For example, under normal circumstances, they generate many `INFO` messages and no `EMERGENCY` messages.

14.2.3 Log Files

By default, Oracle Event Processing server writes log messages to the `server.log` and `consoleoutput.log` files in the */Oracle/Middleware/my_oep/*`user_projects/domains/`*DOMAIN_DIR/servername* directory, where *DOMAIN_DIR* refers to the domain directory (such as `my_domain`), and *servername* refers to the server instance directory (such as `myServer1`).

For information on configuring log file attributes, see log-file.

14.2.4 Log Message Format

Oracle Event Processing server writes log messages in different formats depending on the type of log file it is writing to.

14.2.4.1 Format of Output to a Log File

The system writes a message to the specified log file consisting of a #### prefix, `Timestamp`, `Severity`, `Subsystem`, `Server Name`, `Connection`, `Thread ID` or `User ID` or `Transaction ID`, `Message ID`, and the `Message`, along with a stack trace if any. Each attribute is contained between angle brackets.

The following is an example of a message in the server log file (split for readability; in practice, the message might be on one line):

```
####<May 25, 2015 10:23:32 AM EST> <Notice> <Deployment> <> <myServer>
<RMI TCP Connection(4)-141.144.123.236> <> <> <> <1235575412801> <BEA-2045000>
<The application bundle "Hello" was deployed successfully to file
[C:\Oracle\Middleware\my_oep\user_projects\domains\oep_domain\defaultserver
\applications\Hello\Hello.jar]
with version 1235575412708>
```

14.2.4.2 Format of Output to Console, Standard Out, and Standard Error

The system writes a message to the console, standard out, or standard error consisting of `Locale-formatted Timestamp`, `Severity`, `Subsystem`, `Message ID`, and `Message`.

The following is an example of how the message from the previous section would be printed to standard out (split for readability; in practice, the message might be on one line):

```
<May 25, 2015 10:23:32 AM EST> <Notice> <Deployment> <BEA-2045000>
<The application bundle "Hello" was deployed successfully to file
[C:\Oracle\Middleware\my_oep\user_projects\domains\oep_domain\defaultserver
\applications\Hello\Hello.jar]
with version 1235575412708>
```

14.3 OSGi Framework Logger

Oracle Event Processing has a low-level framework logger that is started before the OSGi framework. It is used to report logging event deep inside the OSGi framework and function as a custom default for the logging subsystem before it is configured.

For example, a user may see some log message, which has lower level or severity than what is set in the `config.xml` but higher or equal to what is set on the Launcher command line on the console or in the log file. Until the logging subsystem has started, log messages come from the framework logger and use the framework logging level to filter messages.

14.4 Log4j Logger

Log4j is an open source tool developed for putting log statements in your application. Log4j has three main components: loggers, appenders, and layouts, which are all described in this section.

The Log4j Java logging facility was developed by the Jakarta Project of the Apache Foundation. See:

- The Log4j Project at http://logging.apache.org/log4j/.
- The Log4j API at http://logging.apache.org/log4j/1.2/apidocs/index.html.
- Short introduction to log4j at http://logging.apache.org/log4j/1.2/manual.html.

14.4.1 Loggers

Log4j defines a `Logger` class. An application can create multiple loggers, each with a unique name. In a typical usage of Log4j, an application creates a `Logger` instance for each application class that will emit log messages. Loggers exist in a name space hierarchy and inherit behavior from their ancestors in the hierarchy.

14.4.2 Appenders

This section explains about defining appenders.

Log4j defines appenders (handlers) to represent destinations for logging output. Multiple appenders can be defined. For example, an application might define an appender that sends log messages to standard out, and another appender that writes log messages to a file. Individual loggers might be configured to write to zero or more appenders. One example usage would be to send all logging messages (all levels) to a log file, but only `ERROR` level messages to standard out.

14.4.3 Layouts

Log4j defines layouts to control the format of log messages. Each layout specifies a particular message format. A specific layout is associated with each appender. This lets you specify a different log message format for standard out than for file output, for example.

14.5 Configure the Logging Service

You configure Oracle Event Processing logging service attributes using Oracle Event Processing Visualizer or by editing the Oracle Event Processing server `config.xml` file.

For more information on configuring logging using Oracle Event Processing Visualizer, see *Using Visualizer for Oracle Event Processing*.

The `config.xml` file is located in the */Oracle/Middleware/my_oep/* `user_projects/domains/`*DOMAIN_DIR*`/`*servername*`/config` directory, where *DOMAIN_DIR* refers to the domain directory (such as `my_domain`), and *servername* refers to the server instance directory (such as `myServer1`).

The following example shows a typical Oracle Event Processing server `config.xml` file with logging elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2007 sp2 (http://www.altova.com)-->
<n1:config
    xsi:schemaLocation="http://www.bea.com/ns/wlevs/config/server
wlevs_server_config.xsd"
    xmlns:n1="http://www.bea.com/ns/wlevs/config/server"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
        <name>myLogService</name>
        <log-file-config>myFileConfig</log-file-config>
        <stdout-config>myStdoutConfig</stdout-config>
        <logger-severity>Notice</logger-severity>
        <logger-severity-properties>
            <entry>
                <key>LifeCycle</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Management</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>CQLProcessor</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>CqlProcessor</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Stream</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Ede</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Cache</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Adapters</key>
```

```
                <value>Notice</value>
            </entry>
            <entry>
                <key>Spring</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Channel</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Recplay</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Monitor</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Server</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>EventTrace</key>
                <value>Notice</value>
            </entry>
            <entry>
                <key>Deployment</key>
                <value>Notice</value>
            </entry>
        </logger-severity-properties>
    </logging-service>
    <log-file>
        <name>myFileConfig</name>
        <rotation-type>none</rotation-type>
    </log-file>
    <log-stdout>
        <name>myStdoutConfig</name>
        <stdout-severity>Debug</stdout-severity>
    </log-stdout>

</n1:config>
```

The following sections provide information on configuring Oracle Event Processing logging:

- logging-service
- log-file
- log-stdout
- Configure Severity for an Individual Module.

14.5.1 logging-service

This section provides information on the `logging-service` element:

Table 14-2 Configuration Parameters for logging-service

Parameter	Type	Description
`name`	`String`	The name of this configuration object.
`log-file-config`	`String`	The configuration of the log file and its rotation policies. See log-file.
`stdout-config`	`String`	The name of the `stdout` configuration object used to configure `stdout` output. See log-stdout.
`logger-severity`	`String`	Defines the threshold importance of the messages that are propagated to the handlers. The default value is `Info`. To see `Debug` and `Trace` messages, configure the `logger-severity` to either `Debug` or `Trace`. Valid values are: `Emergency`, `Alert`, `Critical`, `Error`, `Warning`, `Notice`, `Info`, `Debug`, and `Trace`.
`logger-severity-properties`	One or more `<entry>` child elements.	List of name-value pairs, enclosed in an `<entry>` element, that list individual modules (package name, application name, class name, or component such as `CQLProcessor`) and their logging severity. These severities override the default severity of the Oracle Event Processing server. See Configure Severity for an Individual Module.

14.5.2 log-file

This section provides information on the `log-file` element:

Table 14-3 Configuration Parameters for log-file

Parameter	Type	Description
`name`	`String`	The name of this configuration object.
`base-log-file-name`	`String`	The log file name. Default value is `server.log`.

Parameter	Type	Description
log-file-severity	String	Specifies the least important severity of messages written to the log file. Default value is Trace. Valid values are: • Emergency • Alert • Critical • Error • Warning • Notice • Info • Debug • Trace.
log-file-rotation-dir	String	Specifies the directory where old rotated files are stored. If not set, the old files are stored in the same directory as the base log file.
rotation-type	String	Specifies how rotation is performed based on size, time, or not at all. Valid values are: • bySize • byTime • none
rotation-time	String	The time in k:mm format, where k is the hour specified in 24 hour notation and mm is the minutes. Default is 00:00
rotation-time-span-factor	Long	Factor applied to the timespan to determine the number of milliseconds that becomes the frequency of time based log rotations. Default is 3600000.
rotated-file-count	Integer	Specifies the number of old rotated files to keep if number-of-files-limited is true. Default value is 7.
rotation-size	Integer	The size threshold, in KB, at which the log file is rotated. Default is 500.
rotation-time-span	Integer	Specifies the interval for every time-based log rotation. Default value is 24.
rotate-log-on-startup-enabled	Boolean	If true, the log file is rotated on startup. Default value is true.
number-of-files-limited	Boolean	If true, old rotated files are deleted. Default is false.

14.5.3 log-stdout

This section provides information on the `log-stdout` element:

Table 14-4 Configuration Parameters for log-stdout

Parameter	Type	Description
`name`	`String`	The name of this configuration object.
`stdout-severity`	`String`	The threshold severity for messages sent to stdout. Default value is `Notice`. Valid values are: • `Emergency` • `Alert` • `Critical` • `Error` • `Warning` • `Notice` • `Info` • `Debug` • `Trace`.
`stack-trace-depth`	`Integer`	The number of stack trace frames to display on stdout. A default value of `-1` means all frames are displayed.
`stack-trace-enabled`	`Boolean`	If true, stack traces are dumped to the console when included in logged messages. Default value is `true`.

14.5.4 Configure Severity for an Individual Module

Individual modules of Oracle Event Processing can specify their logging severity. This severity overrides the default logging severity of the Oracle Event Processing server.

You do this by specifying an `entry` child element in the `logger-severity-properties` element in the Oracle Event Processing server `config.xml` file. You can specify multiple `entry` child elements for any number of modules.

Configure Severity for an Individual Module

1. Edit the Oracle Event Processing server `config.xml` file.
2. Add an `entry` child element to the `logger-severity-properties` element as the following example shows.

```
<logging-service>
  <name>myLogService</name>
  <logger-severity>Warning</logger-severity>
  <logger-severity-properties>
  ...
      <entry>
          <key>CQLProcessor</key>
          <value>Debug</value>
      </entry>
```

```
    ...
    </logger-severity-properties>
    ...
  </logging-service>
```

3. Set the `key` element to any of the following:

   - **Component name**: a component name constant exactly as Table 14-5 lists.

   Table 14-5 Logging Component Name Constants

Component Name Constant	Description
`Adapters`	Applies to log messages from adapter instances running on the Oracle Event Processing server.
`Cache`	Applies to log messages from caching systems and cache instances running on the Oracle Event Processing server.
Cartridges	Applies to log messages related to the Oracle Event Processing cartridge infrastructure.
`Channel`	Applies to log messages from channels running on the Oracle Event Processing server.
`CQLProcessor`	Applies to log messages from Oracle CQL processors running on the Oracle Event Processing server.
`CQLServer`	Applies to log messages from the CQLEngine, which is at the core of each CQLProcessor.
`CQLServerTrace`	Applies to log messages from the CQLEngine, which is at the core of each CQLProcessor
`Coherence`	Applies to log messages from Oracle Coherence, including messages related to clustering. The value you enter here is mapped to Oracle Coherence severity levels in the following way: Error: 1 Warning: 2 Notice: 3 Info: 4 Debug: 5 Trace: 9 You can customize logging from Oracle Coherence by overriding the `logging-config` setting in its configuration. For example, you can override the default log destination used for Oracle Coherence (log4j) and use another. For more information on overriding configuration, see Configure the Oracle Coherence Cluster.
Deployment	Applies to log messages related to the deployment infrastructure.
`Ede`	Applies to log messages from the Event-Driven Environment, the Oracle Event Processing server event-dispatching infrastructure.

Component Name Constant	Description

`EventTrace`	When set to `Info` or `Debug`, allows you to trace events as they flow through the EPN for all applications. You can dynamically change the severity of this log key using Oracle Event Processing Visualizer.
	At the `Info` severity, you see log messages like:
	`<May 26, 2009 5:53:49 PM PDT> <Info> <EventTrace> <BEA-000000> <Application [helloworld], Stage [helloworldOutputChannel] received insert event>`
	At the `Debug` severity, the log messages include details of the event:
	`<May 26, 2009 6:02:34 PM PDT> <Debug> <EventTrace> <BEA-000000> <Application [helloworld], Stage [helloworldOutputChannel] received insert event [HelloWorldEvent: HelloWorld - the current time is: 6:02:34 PM]>`

FaultHandler	Applies to log messages related to fault handling in Oracle CQL.
HadoopCartridge	Applies to log messages related to the Hadoop cartridge.
JavaCartridge	Applies to log messages related to the use of Java in Oracle CQL.
`Lifecycle`	Applies to log messages from Oracle Event Processing server and application life cycle operations.
`Management`	Applies to log messages from Oracle Event Processing server general JMX-related management API operations.
`Monitor`	Applies to log messages from the Oracle Event Processing server monitoring service.
NoSQLCartridge	Applies to log messages related to the Oracle NoSQL cartridge.
`Recplay`	Applies to log messages from Oracle Event Processing server event recording and playback operations.
Server	Applies to log messages related to server infrastructure components.
SpatialCartridge	Applies to log messages related to the spatial cartridge.
`Spring`	Applies to log messages from Spring container operations.
`Stream`	Applies to log messages from stream instances running on the Oracle Event Processing server.

For example:

```
<entry>
    <key>CQLProcessor</key>
    <value>Debug</value>
</entry>
```

- **Application name**: the module name of any Oracle Event Processing server or user-defined application. For example:

```
<entry>
    <key>sample.HelloWorld</key>
    <value>Debug</value>
</entry>
```

4. Set the `value` element to a severity level.

   See Use Log Severity Levels.

   For example:

```
<entry>
    <key>CQLProcessor</key>
    <value>Debug</value>
</entry>
```

   This severity level applies to the module you specified in the `key` element and overrides the default Oracle Event Processing server logging severity level set in the `logger-severity` element.

5. Repeat from step 2 for any other modules.

6. Save and close the `config.xml` file.

14.6 Configure Log4j Logging

Oracle Event Processing supports the open-source log4j logging system.

This section describes the following tasks:

- Configure log4j Properties
- Configure Application Manifest
- Enable Log4j Logging
- Debug Log4j Logging.

14.6.1 Configure log4j Properties

The default configuration file is `log4j.properties`. It can be overridden by using the `log4j.configuration` system property. See https://www.qos.ch/shop/products/log4j/log4j-Manual.jsp.

The following is an example of a `log4j.properties` file:

```
log4j.rootLogger=debug, R
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=D:/log4j/logs/mywebapp.log
log4j.appender.R.MaxFileSize=10MB
log4j.appender.R.MaxBackupIndex=10
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%p %t %c - %m%n
log4j.logger=DEBUG, R
```

14.6.2 Configure Application Manifest

Update the `MANIFEST.MF` file of your application to import the following required Log4j packages.

```
Import-Package:
    org.apache.log4j;version="1.2.13",
    org.apache.log4j.config;version="1.2.13",
    ...
```

14.6.3 Enable Log4j Logging

To specify logging to a Log4j `Logger`, set the following system properties on the command line:

```
   -
Dorg.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl
   -Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
   -Dlog4j.configuration=<URL>/log4j.properties
```

Another very useful command line property is `-Dlog4j.debug=true`. Use this property when log4j output fails to appear or you get cryptic error messages.

14.6.4 Debug Log4j Logging

If log4j output fails to appear or you get cryptic error messages, consider using the command line property `-Dlog4j.debug=true` on the command line.

For more information, see Enable Log4j Logging.

14.7 Use the Apache Commons Logging API

You can use Apache Commons logging API in your Oracle Event Processing applications to log application-specific messages to the Oracle Event Processing `server.log` and `consoleoutput.log` files.

To use the commons logging API:

1. Set the system property `org.apache.commons.logging.LogFactory` to `weblogic.logging.commons.LogFactoryImpl`.

   This `LogFactory` creates instances of `weblogic.logging.commons.LogFactoryImpl` that implement the `org.apache.commons.logging.Log` interface.

2. From the `LogFactory`, get a reference to the Commons `Log` object by name.

   This name appears as the subsystem name in the log file.

3. Use the `Log` object to issue log requests to logging services.

   The Commons `Log` interface methods accept an object. In most cases, this will be a string containing the message text.

   The Commons `LogObject` takes a message ID, subsystem name, and a string message argument in its constructor. See `org.apache.commons.logging` at http://jakarta.apache.org/commons/logging/api/index.html.

4. The `weblogic.logging.commons.LogImpl` log methods direct the message to the server log.

```
import org.apache.commons.logging.LogFactory;
import org.apache.commons.logging.Log;

public class MyCommonsTest {
  public void testCommonsLogging() {
    System.setProperty(LogFactory.FACTORY_PROPERTY,
      "weblogic.logging.commons.LogFactoryImpl");
    Log clog = LogFactory.getFactory().getInstance("MyCommonsLogger");
    // Log String objects
    clog.debug("Hey this is common debug");
    clog.fatal("Hey this is common fatal", new Exception());
    clog.error("Hey this is common error", new Exception());
    clog.trace("Dont leave your footprints on the sands of time");
  }
}
```

14.8 Configure Debugging Options

Table 14-6 lists the debugging options that Oracle Event Processing provides. You can enable and disable these debugging options to help diagnose problems with your Oracle Event Processing applications.

Table 14-6 Debug Flags

Debug Flag	Description
`com.bea.core.debug.DebugBootBundle`	Boot Debugging
`com.bea.core.debug.DebugBootBundle.stdout`	Boot Debugging debug strings go to stdout
`com.bea.core.debug.DebugCM`	Configuration Manager
`com.bea.core.debug.DebugCM.stdout`	Configuration Manager debug strings go to stdout
`com.bea.core.debug.DebugConfigurationRuntime`	Runtime information from the Runtime MBeans
`com.bea.core.debug.DebugCSS`	CSS
`com.bea.core.debug.DebugCSS.stdout`	CSS debug strings go to stdout
`com.bea.core.debug.DebugCSSServices`	CSS Services
`com.bea.core.debug.DebugCSSServices.stdout`	CSS Services debug strings go to stdout

Debug Flag	Description
com.bea.core.debug.DebugJDBCConn	JDBC Connection
com.bea.core.debug.DebugJDBCInternal	JDBC Internal
com.bea.core.debug.DebugJDBCRMI	JDBC RMI
com.bea.core.debug.DebugJDBCSQL	JDBC SQL
com.bea.core.debug.DebugJTA2PC	JTA 2PC
com.bea.core.debug.DebugJTA2PCDetail	JTA 2PCDetail
com.bea.core.debug.DebugJTA2PCStackTrace	JTA 2PCStackTrace
com.bea.core.debug.DebugJTAGateway	JTA Gateway
com.bea.core.debug.DebugJTAGatewayStackTrace	JTA GatewayStackTrace
com.bea.core.debug.DebugJTAHealth	JTA Health
com.bea.core.debug.DebugJTAJDBC	JTA JDBC
com.bea.core.debug.DebugJTALifecycle	JTA Lifecycle
com.bea.core.debug.DebugJTALLR	JTA LLR
com.bea.core.debug.DebugJTAMigration	JTA Migration
com.bea.core.debug.DebugJTANaming	JTA Naming
com.bea.core.debug.DebugJTANamingStackTrace	JTA NamingStackTrace
com.bea.core.debug.DebugJTANonXA	JTA NonXA

Debug Flag	Description
com.bea.core.debug.DebugJTAPropagate	JTA Propagate
com.bea.core.debug.DebugJTARecovery	JTA Recovery
com.bea.core.debug.DebugJTAResourceHealth	JTA ResourceHealth
com.bea.core.debug.DebugJTATLOG	JTA TLOG
com.bea.core.debug.DebugJTAXA	JTA XA
com.bea.core.debug.DebugJTAXAStackTrace	JTA XAStackTrace
com.bea.core.debug.DebugNetIO	NetIO
com.bea.core.debug.DebugOX	OSGi to JMX (OX)
com.bea.core.debug.DebugOX.stdout	OSGi to JMX (OX), debug goes to standard out.
com.bea.core.debug.DebugSCP	Simple Configuration Provider
com.bea.core.debug.DebugSCP.stdout	Simple Configuration Provider debug strings go to stdout
com.bea.core.debug.DebugSDS	Simple Declarative Services
com.bea.core.debug.DebugSDS.stdout	SDS debug strings go to stdout
com.bea.core.debug.DebugServiceHelper	Service Helper
com.bea.core.debug.DebugServiceHelper.stdout	Service Helper debug strings go to stdout
com.bea.core.debug.DebugStoreAdmin	Store Administration
com.bea.core.debug.DebugStoreIOLogical	Store IOLogical

Debug Flag	Description
com.bea.core.debug.DebugStoreIOLogicalBoot	Store IOLogicalBoot
com.bea.core.debug.DebugStoreIOPhysical	Store IOPhysical
com.bea.core.debug.DebugStoreIOPhysicalVerbose	Store IOPhysicalVerbose
com.bea.core.debug.DebugStoreXA	Store XA
com.bea.core.debug.DebugStoreXAVerbose	Store XAVerbose
com.bea.core.debug.servicehelper.dumpstack	Dump stack traces when Service Helper times out.

The following sections provide information on how to use these Oracle Event Processing debugging options:

- Configure Debugging Options with System Properties
- Configure Debugging Options with a Configuration File

If you are using Log4j logging, see also Debug Log4j Logging.

14.8.1 Configure Debugging Options with System Properties

Use the following steps to configure debugging using system properties.

In this procedure, you will turn on Simple Declarative Services (SDS) debugging (`com.bea.core.debug.DebugSDS` from Table 14-6) using the Oracle Event Processing server `startwlevs.sh` file.

Configure Debugging Options using System Properties

1. Locate the `DebugSDS` flag in Table 14-6:

   ```
   com.bea.core.debug.DebugSDS
   ```

2. Create a property by prepending `-D` to the flag:

   ```
   -Dcom.bea.core.debug.DebugSDS
   ```

3. Enable this debug flag by setting the property to `true`:

   ```
   -Dcom.bea.core.debug.DebugSDS=true
   ```

4. Start the Oracle Event Processing server using the `startwlevs.sh` with this property:

   ```
   ./startwlevs.sh -Dcom.bea.core.debug.DebugSDS=true
   ```

14.8.2 Configure Debugging Options with a Configuration File

This section describes the procedure to configure debugging options with a Configuration File.

Use the following steps to configure debugging from a configuration file.

In this procedure, you will turn on Simple Declarative Services (SDS) debugging (`com.bea.core.debug.DebugSDS` from Table 14-6) in the Oracle Event Processing server `config.xml` file.

Configure Debugging Options with a Configuration File

1. Locate the `DebugSDS` flag in Table 14-6:

   ```
   com.bea.core.debug.DebugSDS
   ```

2. Create an XML tag by omitting the `com.bea.core.debug.` package name from the flag name:

   ```
   <DebugSDS></DebugSDS>
   ```

3. Edit the Oracle Event Processing server `config.xml` file and add a `debug` element with a `debug-properties` child element as step 5 shows.

4. Add your `DebugSDS` element to the `debug-properties` element as step 5 shows.

5. Enable this debug flag by setting the `DebugSDS` element to `true` as the following example shows.

   ```
   <config>
      <debug>
         <debug-properties>
            <DebugSDS>true</DebugSDS>
         </debug-properties>
      </debug>
   </config>
   ```

6. Set `logger-severity` to `Debug` in the `logging-service` element as step 6 shows.

7. Set `stdout-severity` to `Debug` in the `log-stdout` element as the following example shows.

   ```
   <config>
      <debug>
         <debug-properties>
            <DebugSDS>true</DebugSDS>
         </debug-properties>
      </debug>

      <logging-service>
         <logger-severity>Debug</logger-severity>
         <stdout-config>logStdout</stdout-config>
         <log-file-config>logFile</log-file-config>
      </logging-service>

      <log-file>
        <name>logFile</name>
        <log-file-severity>Debug</log-file-severity>
   ```

```
      <number-of-files-limited>true</number-of-files-limited>
      <rotated-file-count>4</rotated-file-count>
      <rotate-log-on-startup-enabled>true</rotate-log-on-startup-enabled>
    </log-file>

    <log-stdout>
       <name>logStdout</name>
       <stdout-severity>Debug</stdout-severity>
    </log-stdout>
</config>
```

Part V

Command Reference

This part page contains links to the command reference appendices.

Command Reference contains the following appendices:

- wlevs.Admin Command-Line Reference
- Deployer Command-Line Reference
- Security Utilities Command-Line Reference.

A

wlevs.Admin Command-Line Reference

You can use the `wlevs.Admin` to administer Oracle Event Processing, dynamically configure rules for Oracle Continuous Query Language (Oracle CQL) processors, and monitor event latency and throughput.

This appendix includes the following sections:

A.1 Overview of the wlevs.Admin Utility

The `wlevs.Admin` utility is a command-line interface to administer Oracle Event Processing, to dynamically configure the rules for Oracle CQL processors, and to monitor the event latency and throughput of an application. The utility uses JMX to query the configuration and run time MBeans of servers and deployed applications.

The Oracle Event Processing configuration framework enables concurrent changes to the application and server configuration by multiple users. The framework does not use locking to manage this concurrency, but uses optimistic version-based concurrency. This means that two users can always view the configuration of the same object with the intention to update it, but only one user can commit their changes. The

other user get an error when they try to update the same configuration object and must refresh their session to view the updated configuration.

Each `wlevs.Admin` utility command runs in its own transaction, which means that there is an implicit commit after each execution of a command. If you want to batch multiple configuration changes in a single transaction, use JMX directly to make these changes rather than the `wlevs.Admin` utility.

A.2 Configure the wlevs.Admin Utility Environment

Before you can use the `wlevs.Admin` utility, you must configure your environment appropriately.

Configure the wlevs.Admin Utility Environment

1. Configure JMX connectivity for the domain you want to administer. See JMX .
2. Set your `CLASSPATH` in one of the following ways:
   - Implicitly set your `CLASSPATH` by using the `-jar` argument when you run the utility.

     Set the `-jar` argument to the */Oracle/Middleware/my_oep/oep/bin/wlevsadmin.jar* file.

     When you use the `-jar` argument, you do not specify the `wlevs.Admin` utility name at the command line. For example:

     ```
     java -jar d:/Oracle/Middleware/my_oep/oep/bin/wlevsadmin.jar
          -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
          UPLOAD -application helloworld -processor helloworldProcessor
          -sourceURL file:///d:/test/newrules2.xml
     ```

   - Explicitly update your `CLASSPATH` by adding the following files to the `CLASSPATH` environment variable:

     ```
     /Oracle/Middleware/my_oep/oep/bin/wlevsadmin.jar
     /Oracle/Middleware/my_oep/oep/bin/wlevs.jar
     /Oracle/Middleware/my_oep/oep/modules/
     com.bea.wlevs.deployment.server_12.1.2.0_0.jar
     /Oracle/Middleware/my_oep/oep/modules/com.bea.wlevs.ede_12.1.2.0_0.jar
     /Oracle/Middleware/my_oep/oep/modules/
     com.bea.wlevs.management_12.1.2.0_0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/
     com.bea.core.jndi.context_8.0.0.0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/com.bea.core.jmx_8.0.0.0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/com.bea.core.rmi_8.0.0.0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/
     com.bea.core.i18n_3.0.0.0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/
     com.bea.core.diagnostics.core_4.0.0.0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/javax.xml.stream_1.1.1.0.jar
     /Oracle/Middleware/my_oep/wlserver/modules/
     com.bea.core.bootbundle_13.0.0.0.jar
     ```

A.3 Running the wlevs.Admin Utility Remotely

You can run the `wlevs.Admin` utility on a computer that is different from the computer on which Oracle Event Processing is installed and running.

Run the wlevs.Admin utility Remotely

1. Copy the following JAR files from the computer on which Oracle Event Processing is installed to the computer on which you want to run `wlevs.Admin`; you can copy the JAR files to the directory name of your choice:

   /Oracle/Middleware/my_oep/`oep/bin/wlevsadmin.jar`
   /Oracle/Middleware/my_oep/oep/bin/wlevs.jar
   `/Oracle/Middleware/my_oep/`*oep/modules/*
   com.bea.wlevs.deployment.server_12.1.2.0_0.jar
   /Oracle/Middleware/my_oep/oep/modules/com.bea.wlevs.ede_12.1.2.0_0.jar
   /Oracle/Middleware/my_oep/oep/modules/
   com.bea.wlevs.management_12.1.2.0_0.jar
   /Oracle/Middleware/my_oep/wlsever/modules/
   com.bea.core.jndi.context_8.0.0.0.jar
   /Oracle/Middleware/my_oep/wlserver/modules/com.bea.core.jmx_8.0.0.0.jar
   /Oracle/Middleware/my_oep/wlserver/modules/com.bea.core.rmi_8.0.0.0.jar
   /Oracle/Middleware/my_oep/wlserver/modules/com.bea.core.i18n_3.0.0.0.jar
   /Oracle/Middleware/my_oep/wlserver/modules/
   com.bea.core.diagnostics.core_4.0.0.0.jar
   /Oracle/Middleware/my_oep/wlserver/modules/javax.xml.stream_1.1.1.0.jar

2. Set your `CLASSPATH` in one of the following ways:

   - Implicitly set your `CLASSPATH` by using the `-jar` argument when you run the utility. Set the argument to the *`NEW_DIRECTORY`*`/wlevsadmin.jar` file, where *`NEW_DIRECTORY`* refers to the directory on the remote computer into which you copied the required JAR files. When you use the `-jar` argument, you do not specify the `wlevs.Admin` utility name at the command line.

   - Explicitly update your `CLASSPATH` by adding all the files you copied to the remote computer to your `CLASSPATH` environment variable:

3. Run the `wlevs.Admin` utility in SSL mode as described in Run wlevs.Admin Utility in SSL Mode.

A.4 Run wlevs.Admin Utility in SSL Mode

To use SSL when using the `wlevs.Admin` command-line utility, you must first create a trust key store. For more information, see SSL to Secure Network Traffic.

Run wlevs.Admin utility in SSL Mode

1. If not already running, start the Oracle Event Processing server.

   See Start and Stop Servers.

2. Change to `ssl` directory.

   By default the ssl directory is in `Oracle/Middleware/my_oep/user_projects/domains/<domainname>/<servername>`.

3. Generate a trust key store by specifying the following command on one line:

   ```
   java -classpath Oracle\Middleware\my_oep\oep\
   common\lib\evspath.jar;Oracle\Middleware\my_oep\oep\utils\security\
   wlevsgrabcert.jar com.bea.wlevs.security.util.GrabCert host:secureport
   -alias=alias truststorepath
   ```

host: The computer on which `myServer2` is running.

secureport: The SSL network I/O port configured for `myServer2`. The default value is 9003. For more information, see Configure SSL Manually.

alias: The alias for the certificate in the trust key store. Default value is the host name.

truststorepath: The full path name of the generated trust key store file. The default is `evstrust.jks`.

For example:

```
java -classpath C:\Oracle\Middleware\my_oep\oep\
common\lib\evspath.jar;C:\Oracle\Middleware\my_oep\utils\security\
wlevsgrabcert.jar com.bea.wlevs.security.util.GrabCert myServer2:9003 -
alias=myServer2 evstrust.jks
```

4. Use the following properties and specify the secure port in the URL as shown in the following example.

   `-Djavax.net.ssl.trustStore`: The name of the trust key store file you created in the preceding step.

   `-Djavax.net.ssl.trustStorePassword`: The password of the trust key store file.

```
java -Djavax.net.ssl.trustStore=clitrust.jks
     -Djavax.net.ssl.trustStorePassword=secret
     -jar wlevsadmin.jar
     -url service:jmx:msarmis://localhost:9003/jndi/jmxconnector
     -username wlevs -password wlevs
     SHUTDOWN -scheduleAt 600
```

A.5 Syntax for Calling the wlevs.Admin Utility

The syntax for using the `wlevs.Admin` utility is as follows:

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    COMMAND-NAME command-arguments
```

The command names and arguments are not case sensitive.

The following sections provide detailed syntax information about the arguments you can supply to the `wlevs.Admin` utility:

- Connection Arguments
- User Credentials Arguments
- Common Arguments

The following sections provide detailed syntax information about the supported commands of the `wlevs.Admin` utility:

- HELP Command
- SHUTDOWN Command
- Commands to Manage Oracle CQL Rules

- Commands to Manage MBeans
- Commands for Controlling Event Record and Playback
- Commands for Monitoring Throughput and Latency.

A.5.1 Example Environment

In many of the examples in the following sections the environment has the following setup:

- The Oracle Event Processing server instance listens to JMX requests on port `9002`.
- The Oracle Event Processing server instance is installed on a host machine named `ariel` and uses this host name for its listen address.
- The `wlevs` user name has system administrator privileges and uses `wlevs` for the administrator password.

Also, all of the examples are shown on multiple lines. When you run the command, enter all arguments and commands on a single line.

A.5.2 Exit Codes Returned by wlevs.Admin

All `wlevs.Admin` commands return an exit code of `0` when the command succeeds and an exit code of `1` when the command fails.

To view the exit code from a Windows command prompt, enter `echo%ERRORLEVEL%` after you run a `wlevs.Admin` command. To view the exit code in a `bash` shell, enter `echo $?`.

`wlevs.Admin` calls `System.exit(1)` when an exception is raised while processing a command, which causes Ant and other Java client JVMs to exit.

A.6 Connection Arguments

```
java wlevs.Admin
    [ {-url URL} | -protocol protocol | {-listenAddress hostname -listenPort port} ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    COMMAND-NAME command-arguments
```

When you invoke most `wlevs.Admin` commands, you must specify the arguments in Table A-1 to connect to an Oracle Event Processing server instance.

Table A-1 Connection Arguments

Argument	Definition
`-url service:jmx:msarmi://`*`host`*`:`*`port`*`/jndi/jmxconnector`	Specifies the URL that establishes a JMX connection to the Oracle Event Processing instance that you want to administer, where: • *`host`* refers to the name of the computer on which the Oracle Event Processing instance is running • *`port`* refers to the Oracle Event Processing server JNDI port If you use this argument, do not specify `-listenAddress` or `-listenPort`. Other than *`host`*, you specify the remainder of the URL as written. For example, if Oracle Event Processing is running on a computer with host name ariel, and the JMX listening port is 9002, then the URL would be: `-url service:jmx:msarmi://ariel:9002/jndi/jmxconnector` See JMX for details about configuring JMX, JNDI, and RMI for Oracle Event Processing.
`-listenAddress` *`hostname`*	Specifies the name of computer on which the Oracle Event Processing instances is running. This argument, together with `-listenPort`, is used to build the URL that establishes a JMX connection to the server you want to administer. You use this argument, together with `-listenPort`, *instead* of `-url`. For example, if Oracle Event Processing is running on a computer with host name `ariel`, then this argument would be: `-listenAddress ariel`

`-listenPort` *`port`*	Specifies the port configured for Oracle Event Processing that listens to JMX connections. This argument, together with `-listenAddress`, is used to build the URL that establishes a JMX connection to the server you want to administer.
	You use this argument, together with `-listenAddress`, *instead* of `-url`.
	The JMX port is configured in the `config.xml` file of the Oracle Event Processing domain you are administering. In particular, the port is the `<port>` child element of the `<netio>` element, as shown:
	`<netio>`
	`<name>NetIO</name>`
	`<port>`**`9002`**`</port>`
	`</netio>`
	In the example, the port is `9002` and you specify as an argument as follows:
	`-listenPort 9002`
	See JMX for details about configuring JMX, JNDI, and RMI for Oracle Event Processing.

A.7 User Credentials Arguments

```
java wlevs.Admin
    [ Connection Arguments ]
    [ -username username [-password password] ]
    [ Common Arguments ]
    COMMAND-NAME command-arguments
```

When you invoke most `wlevs.Admin` commands, you must specify the arguments in Table A-2 to provide the user credentials of an Oracle Event Processing user who has permission to invoke the command. If security has not been enabled for your Oracle Event Processing domain, then you do not have to provide user credentials.

Table A-2 User Credentials Arguments

Argument	Definition
-username *username*	The name of the user who is issuing the command. This user must have appropriate permission to view or modify the target of the command.
-password *password*	The password that is associated with the username.

Note:

The exit code for all commands is 1 when the `wlevs.Admin` utility cannot connect to the server or when the Oracle Event Processing server instance rejects the user name and password.

A.8 Common Arguments

This section lists the common arguments.

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ -verbose ]
    COMMAND-NAME command-arguments
```

All `wlevs.Admin` commands support the argument in Table A-3 to get verbose output.

Table A-3 Common Arguments

Argument	Definition
-verbose	Specifies that `wlevs.Admin` should output additional verbose information.

A.9 HELP Command

Provides syntax and usage information for all Oracle Event Processing commands or for a single command if a command value is specified on the HELP command line. You can issue this command from any computer on which Oracle Event Processing is installed. You do not need to start a server instance to invoke this command, nor do you need to supply user credentials, even when security is enabled for the server.

Syntax

```
java wlevs.Admin HELP [COMMAND]
```

The *COMMAND* argument can be:

- The keyword ALL, which returns usage information about all commands.
- One of the keywords MBEAN, RULES, or LIFECYCLE, which returns usage information about the three different groups of commands.
- A command such as UPLOAD, which returns usage information about the particular command.

Example

In the following example, information about using the UPLOAD command is requested:

```
java wlevs.Admin HELP UPLOAD
```

The command returns the following output:

```
Description:
Uploads rules to be configured in the processor.

Usage:
java wlevs.Admin
   [-url | -listenAddress <host-name> -listenPort <port>]
  -username <username> -password <password>
   UPLOAD -application <application name> -processor <processor name> -sourceURL "source url"

Where:
-application = Name of the application.
-processor = Name of the processor.
-sourceURL = source URL containing the rules in an XML format.

java wlevs.Admin -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs UPLOAD -application myapplication
     -processor processor -sourceURL file:/d:/test/rules.xml
```

A.10 SHUTDOWN Command

The SHUTDOWN command manages the life cycle of a server instance by gracefully shutting down the specified Oracle Event Processing server instance. A graceful shutdown gives Oracle Event Processing time to complete certain application processing currently in progress.

The -url connection argument specifies the particular Oracle Event Processing server instance that you want to shut down, based on the host and jmxport values. See Connection Arguments for details.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    SHUTDOWN [-scheduleAt seconds]
```

Table A-4 SHUTDOWN Arguments

Argument	Definition
`-scheduleAt` *seconds*	Specifies the number of seconds after which the Oracle Event Processing instance shuts down. If you do not specify this parameter, the server instance shuts down immediately.

Example

The following example instructs the specified Oracle Event Processing instance to shut down in ten minutes:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     SHUTDOWN -scheduleAt 600
```

After you issue the command, the server instance prints messages to its log file and to its standard out. The messages indicate that the server state is changing and that the shutdown sequence is starting.

A.11 Commands to Manage Oracle CQL Rules

This section lists the commands to manage Oracle CQL Rules.

Table A-5 is an overview of commands that manage the Oracle CQL rules for a particular processor of an Oracle Event Processing application.

Table A-5 Overview of Commands for Managing Application Oracle CQL Rules

Command	Description
GETRULE	Returns the text of an existing Oracle CQL rule, query, or view of the processor of an Oracle Event Processing application.
ADDRULE	Adds a new Oracle CQL rule, query, or view to the processor of an Oracle Event Processing application.
DELETERULE	Deletes an existing Oracle CQL rule, query, or view from the processor of an Oracle Event Processing application.
REPLACERULE	Replaces an existing Oracle CQL rule, query, or view with new Oracle CQL text.
STARTRULE	Starts a previously stopped Oracle CQL rule or query.
STOPRULE	Stops a previously started Oracle CQL rule or query.
UPLOAD	Configures a set of Oracle CQL rules, queries, or views for a processor of an Oracle Event Processing application by uploading the rules from a component configuration XML file.
DOWNLOAD	Downloads the set of Oracle CQL rules, queries, or views associated with a processor of an Oracle Event Processing application to a component configuration XML file.

A.11.1 GETRULE

Returns the full text of an Oracle CQL rule, query, or view from the specified Oracle CQL processor of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    GETRULE -application application -processor processor -rule rulename
```

Table A-6 GETRULE Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor` *processor*	Specifies the name of the particular Oracle CQL processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-rule` *rulename*	Specifies the name of the Oracle CQL rule, query, or view you want to see. See Query for Application and Processor Names for details on querying for the rule, query, or view name if you do not know it. You can also use the DOWNLOAD command to get the list of rules for a particular processor.

Example

The following example shows how to get the full text of the Oracle CQL view called `myview` from the Oracle CQL `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     GETRULE -application helloworld -processor helloworldProcessor
     -rule myview
```

A.11.2 ADDRULE

This section explains the `ADDRULE` command.

Adds a new Oracle CQL rule, query, or view to the specified processor of an Oracle Event Processing application. If a rule, query, or view with the same name (identified with the *`rulename`*, *`queryname`*, or *`viewname`* parameter) already exists, then the `ADDRULE` command replaces the existing rule, query, or view with the new one.

Note:

An Oracle CQL query immediately begins to output events when its input channels provides input events. If you plan to use a query selector on a channel with an upstream Oracle CQL processor, then you might observe unwanted query results on the downstream channel between the time you add the query to the upstream Oracle CQL processor and the time you configure the query selector on the downstream channel. See *Using Visualizer for Oracle Stream Explorer*.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    ADDRULE -application application -processor processor
        -rule  [rulename]  rulestring  |
        -query [queryname] querystring |
        -view  [viewname]  viewstring [-schema comma-separated-names]
        [-active true | false]
```

Table A-7 ADDRULE Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor` *processor*	Specifies the name of the particular processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-rule [`*rulename*`]` *rulestring*	Specifies the Oracle CQL rule you want to add to the specified processor of your application. An Oracle CQL rules include: • `REGISTER|CREATE FUNCTION` (aggregate and single-row functions) • `REGISTER|CREATE WINDOW` The *`rulename`* parameter is not required; if you do not specify it, Oracle Event Processing generates a name for you. Enter the Oracle CQL *`rulestring`* using double quotes.

Argument	Definition
`-query [queryname] querystring`	Specifies the Oracle CQL query you want to add to the specified processor of your application. The *queryname* parameter is not required; if you do not specify it, Oracle Event Processing generates a name for you. Enter the Oracle CQL *querystring* using double quotes.
`-view [viewname] viewstring [-schema comma-separated-names]`	Specifies the Oracle CQL view you want to add to the specified processor of your application. The *viewname* parameter is not required; if you do not specify it, Oracle Event Processing generates a name for you. Enter the Oracle CQL *viewstring* using double quotes. The *comma-separated-names* parameter is not required; if you do not specify it, Oracle Event Processing generates the schema based on the select statement in the *viewstring*.
`-active true | false`	Specifies if the rule should be started and ready to process events after being added. Valid values for this argument are `true` (start rule after adding) or `false` (do not start rule after adding); default value is `true`. If set to `false`, use STARTRULE to start the rule.

Example

The following example shows how to add the Oracle CQL query `SELECT * FROM Withdrawal [Rows 5]`, with name `myquery`, to the Oracle CQL processor `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     ADDRULE -application helloworld -processor helloworldProcessor
     -query myquery "SELECT * FROM Withdrawal [Rows 5]"
```

A.11.3 DELETERULE

Deletes an existing Oracle CQL rule from the specified processor of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    DELETERULE -application application -processor processor -rule rulename
```

Table A-8 DELETERULE Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor processor`	Specifies the name of the particular processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules, queries, and views you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-rule rulename`	Specifies the name of the Oracle CQL rule, query, or view you want to delete. See Query for Application and Processor Names for details on querying for the rule name if you do not know it. You can also use the DOWNLOAD command to get the list of rules, queries, or views for a particular Oracle CQL processor.

Example

The following example shows how to delete the Oracle CQL view called `myview` from the Oracle CQL `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     DELETERULE -application helloworld -processor helloworldProcessor -rule myview
```

A.11.4 REPLACERULE

Replaces an existing Oracle CQL rule, query, or view with another rule, query, or view. Oracle Event Processing first destroys the original rule, query, or view and then inserts the new one in its place.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    REPLACERULE -application application -processor processor
        -rule rulename rulestring
```

Table A-9 REPLACERULE Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor processor`	Specifies the name of the particular Oracle CQL processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-rule rulename rulestring`	Specifies the name of the Oracle CQL rule, query, or view you want to replace. Oracle Event Processing deletes the old rule, query, or view and then inserts a new one, with the same name but with the new rule text. In the case of a view, Oracle Event Processing generates the schema based on the select statement in the *`rulestring`*. Enter the Oracle CQL *`rulestring`* using double quotes.

Example

The following example shows how to replace an Oracle CQL query called `myquery` with the Oracle CQL text `SELECT * FROM Withdrawal [Rows 10]` in the Oracle CQL `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     REPLACERULE -application helloworld -processor helloworldProcessor
     -rule myquery "SELECT * FROM Withdrawal [Rows 10]"
```

A.11.5 STARTRULE

Starts an existing Oracle CQL rule or query that was previously stopped in the specified processor of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    STARTRULE -application application -processor processor -rule rulename
```

Table A-10 STARTRULE Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor processor`	Specifies the name of the particular processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules, queries, and views you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-rule rulename`	Specifies the name of the Oracle CQL rule or query you want to start. **NOTE**: You cannot stop and start a view. Views are always active. See Query for Application and Processor Names for details on querying for the rule name if you do not know it. You can also use the DOWNLOAD command to get the list of rules, queries, or views for a particular Oracle CQL processor.

Example

The following example shows how to start the Oracle CQL query called `myquery` from the Oracle CQL `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     STARTRULE -application helloworld -processor helloworldProcessor -rule myquery
```

A.11.6 STOPRULE

Stops an existing Oracle CQL rule or query that was previously started in the specified processor of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    STOPRULE -application application -processor processor -rule rulename
```

Table A-11 STOPRULE Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules or queries you want to manage. See Query for Application and Processor Names for details on using `wlevs.Admin` to get the exact name of your application if you do not currently know it. You can also get the exact application name by looking at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor` *processor*	Specifies the name of the particular processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules, queries, and views you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-rule` *rulename*	Specifies the name of the Oracle CQL rule or query you want to stop. **NOTE**: You cannot stop and start a view. Views are always active. See Query for Application and Processor Names for details on querying for the rule name if you do not know it. You can also use the DOWNLOAD command to get the list of rules, queries, or views for a particular Oracle CQL processor.

Example

The following example shows how to stop the Oracle CQL query called `myquery` from the Oracle CQL `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     STOPRULE -application helloworld -processor helloworldProcessor -rule myquery
```

A.11.7 UPLOAD

Replaces the configured Oracle CQL rules for a specified processor with the Oracle CQL rules from an uploaded component configuration file.

The component configuration file that contains the list of Oracle CQL rules conforms to the component configuration file schema (see *Schema Reference for Oracle Stream Explorer*). This file contains one or more Oracle CQL rules that replace those currently configured for the specified processor. An example of such a component configuration file follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>
  <processor>
  <name>helloworldProcessor</name>
    <rules>
      <query id="helloworldRule1">
          <![CDATA[  SELECT * FROM HelloWorldEvent [Rows 2] >
      </query>
    </rules>
  </processor>
</config>
```

In the preceding example, the component configuration file configures a single Oracle CQL query, with name `helloworldRule1`, and its Oracle CQL query text is `SELECT * FROM HelloWorldEvent [Rows 2]`.

Caution:

When you use the `UPLOAD` command of the `wlevs.Admin` utility, use the `-processor` argument to specify the name of the Oracle CQL processor to which you want to add the Oracle CQL rules, as you do with the other Oracle CQL commands. This means that the utility *ignores* any `<name>` elements in the component configuration file to avoid any naming conflicts.

For details about and examples of creating the component configuration file, *Developing Applications for Event Processing with Oracle Stream Explorer*.

You can obtain a copy of a processor's component configuration file using the `DOWNLOAD` command as DOWNLOAD describes.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    UPLOAD -application application -processor processor -sourceURL sourcefileURL
```

Table A-12 UPLOAD Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor` *processor*	Specifies the name of the particular Oracle CQL processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-sourceURL` *sourcefileURL*	Specifies the URL of the component configuration file that contains the Oracle CQL rules in the form: `file:///path-to-file`

Example

The following example shows how upload the Oracle CQL rules from the `c:\processor\config\myrules.xml` file to the Oracle CQL `helloworldProcessor` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     UPLOAD -application helloworld -processor helloworldProcessor
     -sourceURL file:///c:/processor/config/myrules.xml
```

A.11.8 DOWNLOAD

Downloads the set of Oracle CQL rules associated with the specified Oracle CQL processor of an Oracle Event Processing application to an XML component configuration file.

The XML file is of the same format as described in UPLOAD.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    DOWNLOAD -application application -processor processor
    -file destinationfile [-overwrite overwrite]
```

Table A-13 DOWNLOAD Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-processor` *processor*	Specifies the name of the particular processor, attached to the Oracle Event Processing application specified with the `-application` argument, whose Oracle CQL rules you want to manage. See Query for Application and Processor Names for details on getting the exact name if you do not know it.
`-file` *destinationfile*	Specifies the name of the component configuration XML file to which you want the `wlevs.Admin` utility to download the Oracle CQL rules. Be sure you specify the full pathname of the file.

Argument	Definition
`-overwrite` *overwrite*	Specifies whether the `wlevs.Admin` utility should overwrite an existing file. Valid values for this argument are `true` or `false`; default value is `false`.

Example

The following example shows how download the set of Oracle CQL rules currently attached to the Oracle CQL `helloworldProcessor` of the `helloworld` application to the file `c:\processor\config\myrules.xml`; the utility overwrites any existing file:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     DOWNLOAD -application helloworld -processor helloworldProcessor
     -file c:\processor\config\myrules.xml -overwrite true
```

A.12 Commands to Manage MBeans

This section lists the commands to manage MBeans.

The following sections describe `wlevs.Admin` commands for managing Oracle Event Processing MBeans.

- Specifying MBean Types
- MBean Management Commands

See the *Java API Reference for Oracle Stream Explorer* for the full description of the Oracle Event Processing MBeans.

A.12.1 Specifying MBean Types

To specify which MBean or MBeans you want to access, view, or modify, all of the MBean management commands require either the `-mbean` argument or the `-type` argument.

Use the `-mbean` argument to operate on a single instance of an MBean.

Use the `-type` argument to operate on all MBeans that are an instance of a type that you specify. An MBean's type refers to the interface class of which the MBean is an instance. All Oracle Event Processing MBeans are an instance of one of the interface classes defined in the `com.bea.wlevs.management.configuration`, `com.bea.wlevs.management.runtime`, `com.bea.wlevs.deployment.mbean` and `com.bea.wlevs.server.management.mbean` packages. For a complete list of all Oracle Event Processing MBean interface classes, see the *Java API Reference for Oracle Stream Explorer* for the respective packages.

To determine the value that you provide for the `-type` argument, do the following: Find the MBean's interface class and remove the `MBean` suffix from the class name. For example, for an MBean that is an instance of the `com.bea.wlevs.management.configuration.CQLProcessorMBean`, use `CQLProcessor`.

A.12.2 MBean Management Commands

Table A-14 is an overview of the MBean management commands.

Table A-14 MBean Management Command Overview

Command	Description
GET	Displays properties of MBeans.
INVOKE	Invokes management operations that an MBean exposes for its underlying resource.
QUERY	Searches for MBeans whose `ObjectName` matches a pattern that you specify.
SET	Sets the specified property values for the named MBean instance.

A.12.3 GET

Displays MBean properties and JMX object names in the format described at.http://docs.oracle.com/javase/7/docs/api/javax/management/ObjectName.html

The output of the command follows:

```
{MBeanName object-name {property1 value} {property2 value}. . .}
. . .
```

Note that the properties and values are expressed as name-value pair where each pair is returned within curly brackets. This format facilitates output parsing by a script.

If `-pretty` is specified, each property-value pair is displayed on a new line and curly brackets are not used to separate the pairs:

```
MBeanName: object-name
property1: value
property2: value
.
.
.
MBeanName: object-name
property1: value
abbribute2: value
```

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    GET [-pretty] {-type mbeanType| -mbean objectName} [-property property1] [-property
property2]...
```

Table A-15 GET Arguments

Argument	Definition
`-type mbeanType`	Returns information for all MBeans of the specified type. For more information, see Specifying MBean Types.
`-mbean objectName`	Fully qualified object name of an MBean in the http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html format. For example, if you want to look up an MBean for a processor stage, the naming is as follows (in practice, the string should be on one line): `"com.bea.wlevs:Name=<name of the Stage>,Type=<type of Mbean>, Application=<name of the application>"`

Argument	Definition
`-pretty`	Places property-value pairs on separate lines.
`-property property`	The name of the MBean property (attribute) or properties to be listed. If `-property` is not specified, all properties are displayed.

Example

The following example displays all properties of the `CQLProcessorMBean` that was registered for the Processor Stage when the application called `helloworld` was deployed in Oracle Event Processing.

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     GET -pretty
     -mbean com.bea.wlevs:Name=cqlprocessor,Type=CQLProcessor,Application=helloworld
```

The following example displays all instances of all `CQLProcessorMBean` MBeans.

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     GET -pretty -type CQLProcessor
```

A.12.4 INVOKE

Invokes a management operation for one or more MBeans. For Oracle Event Processing MBeans, you usually use this command to invoke operations other than the `get`*`Attribute`* and `set`*`Attribute`* that most Oracle Event Processing MBeans provide.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    INVOKE {-type mbeanType | -mbean objectName} -method methodname [argument . . .]
```

Table A-16 INVOKE Arguments

Arguments	Definition
`-type mbeanType`	Invokes the operation on all MBeans of a specific type. For more information, see Specifying MBean Types.
`-mbean objectName`	Fully qualified object name of an MBean in the http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html format. For example, if you want to invoke an MBean for an processor stage, the naming is as follows `"com.bea.wlevs:Name=<name of the Stage>,Type=<type of Mbean>, Application=<name of the application>"`
`-method methodname`	Name of the method to be invoked.

Arguments	Definition
argument	Arguments to be passed to the method call. When the argument is a String array, the arguments must be passed in the following format: "*String1*;*String2*;. . . "

Example

The following example invokes the `addRule` method of MBean `com.bea.wlevs.management.configuration.CQLProcessorMBean`:

```
java wlevs.Admin
    -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
    -username wlevs -password wlevs
    INVOKE -mbean com.bea.wlevs:Name=cqlprocessor,Type=CQLProcessor,Application=helloworld
    -method addRule "SELECT * FROM Withdrawal RETAIN ALL"
```

A.12.5 QUERY

Searches for Oracle Event Processing MBeans where the pattern matches the pattern that you specify according to the `ObjectName` class conventions at: http://docs.oracle.com/javase/1.5.0/docs/api/javax/management/ObjectName.html.

All MBeans that are created from an Oracle Event Processing MBean type are registered in the MBean Server under a name that conforms to the `ObjectName` class conventions. You must know the `ObjectName` of an MBean to use `wlevs.Admin` commands to retrieve or modify specific MBean instances.

The output of the command follows:

```
{MBeanName object-name {property1 value} {property2 value}. . .}
. . .
```

The properties and values are expressed as name-value pairs, each of which is returned within curly brackets. This format facilitates parsing of the output by a script.

If `-pretty` is specified, each property-value pair is displayed on a new line and curly brackets are not used to separate the pairs:

```
MBeanName: object-name
property1: value
property2: value
.
.
.
MBeanName: object-name
property1: value
abbribute2: value
```

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    QUERY -pretty -pattern object-name-pattern
```

Table A-17 QUERY Arguments

Argument	Definition
`-pretty`	Places property-value pairs on separate lines.
`-pattern` *`object-name-pattern`*	A partial http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html for which the `QUERY` command searches. The value must conform to the following pattern: *`property-list`* where property-list specifies one or more components (property-value pairs) of a http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html. You can specify these property-value pairs in any order. Within a given naming property-value pair, there is no pattern matching. Only complete property-value pairs are used in pattern matching. However, you can use the * wildcard character in the place of one or more property-value pairs. For example, `type=epl*` is not valid, but `type=CQLProcessor,*` is valid. If you provide at least one property-value pair in the *`property-list`*, you can locate the wildcard anywhere in the given pattern, provided that the *`property-list`* is still a comma-separated list.

Example

The following example searches for all `com.bea.wlevs.management.configuration.CQLProcessorMBean` MBeans:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     QUERY -pattern *:Type=CQLProcessor,*
```

If the command succeeds, it returns the attributes of the MBeans found (lines broken here for readability):

```
{MBeanName="com.bea.wlevs:Name=MonitorProcessor,Type=CQLProcessor,Application=com.bea.wlevs.data
services"{
 AllRules=defaultRule = select * from DSMonitorEvent retain 1 event where metric > 10000}{
 AllRulesInfo=defaultRule = {RULE_TYPE=RULE, STARTED=true, VALUE=select * from DSMonitorEvent
 retain 1 event where metric > 10000, ID=defaultRule}}{Databases=}{Name=MonitorProcessor}
 {NotificationInfo=[Ljavax.management.MBeanNotificationInfo;@20d319}
 {ObjectName=com.bea.wlevs:Name=MonitorProcessor,Type=CQLProcessor,
 Application=com.bea.wlevs.dataservices}{PlaybackConfiguration=}{PlayingBack=false}
 {RecordConfiguration=}{RecordPlayback=com.bea.wlevs:Name=MonitorProcessor,
 Type=RecordPlayback,Application=com.bea.wlevs.dataservices}{Recording=false}
 {Type=CQLProcessor}}
```

A.12.6 Query for Application and Processor Names

All the commands for managing the CQL rules of an Oracle Event Processing application require you to know the name of the application, and the particular processor to which you want to apply the rules. Typically you know these names, but if you do not, you can use the `QUERY` command to get the information from the MBean instances that represent applications and their attached processors.

In particular, use the following `-pattern argument` to get a list of all applications, processors, and rules for a given Oracle Event Processing instance:

```
-pattern  com.bea.wlevs:*,Type=CQLProcessor
```

For example:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     QUERY -pretty
     -pattern com.bea.wlevs:*,Type=CQLProcessor
```

A sample output of this command is shown below:

```
Command Output
------------------------------------------------------
MBeanName: "com.bea.wlevs:Name=helloworldProcessor,Type=CQLProcessor,Application=helloworld,"
       AllRules:
       helloworldRule =  select * from HelloWorldEvent retain 1 event
--end of command output --------
```

In the sample output above:

- The name of the application is `helloworld`.
- The helloworld application has a processor called `helloworldProcessor`.
- The `helloworldProcessor` has a rule called `helloworldRule`.

A.12.7 SET

Sets the specified property (attribute) values for an MBean.

If the command is successful, it returns `OK` and saves the new values to the server configuration.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    SET {-type mbeanType | -mbean objectName}
    -property property1 property1_value
    [-property property2 property2_value] . . .
```

Table A-18 SET Arguments

Argument	Definition
`-type mbeanType`	Sets the properties for all MBeans of a specific type. For more information, see Specifying MBean Types.
`-mbean objectName`	Fully qualified object name of an MBean in the http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html format: `"com.bea.wlevs:Name=<name of the stage>,Type=<MBean type>,Application=<name of the deployed application>"`
`-property property`	The name of the property to be set.

Argument	Definition

property _value	The value to be set.
	• Some properties require you to specify the name of an Oracle Event Processing MBean. In this case, specify the fully qualified object name of an MBean in the http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html format. For example (in practice, the string should be on one line):
	`"com.bea.wlevs:Name=<name of the stage>,Type=<type of MBean>, Application=<name of the application>"`
	• When the property value is an MBean array, separate each MBean object name by a semicolon and surround the entire property value list with quotes. For example:
	`"com.bea.wlevs:Application=<name of the application>,Type=<type of MBean>,Name=<name of the Stage>;Type=<type of MBean>,Name=<name of the stage>"`
	• When the property value is a String array, separate each string by a semicolon and surround the entire property value list with quotes:
	`"String1;String2;. . . "`
	• When the property value is a String or String array, you can set the value to null by using either of the following:
	`-property property-name ""`
	`-property property-name`
	• If the property value contains spaces, surround the value with quotes:
	`"-Da=1 -Db=3"`

Example

The following example shows how to set the `MaxSize` property of the channel named `helloworldOutstream` of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     SET -mbean com.bea.wlevs:Name=helloworldOutstream,Type=Channel,Application=helloworld
     -property  MaxSize 1024
```

A.13 Commands for Controlling Event Record and Playback

This section explains the commands for controlling event record and playback.

Table A-19 is an overview of commands for managing event record and playback for a particular stage of an Oracle Event Processing application. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Note:

Before you can use commands to control event record and playback on a stage, you must configure the stage with the appropriate event record and playback options. See:

- CONFIGURERECORD
- *Developing Applications for Event Processing with Oracle Stream Explorer*
- *Schema Reference for Oracle Stream Explorer*

Table A-19 Overview of Commands for Controlling Event Record and Playback

Command	Description
STARTRECORD	Starts the recording of events for a stage in an Oracle Event Processing application.
STOPRECORD	Stops the recording of events for a stage in an Oracle Event Processing application.
CONFIGURERECORD	Configures the parameters for the event recording of a stage in an Oracle Event Processing application.
SCHEDULERECORD	Schedules the recording of events for a stage in an Oracle Event Processing application.
LISTRECORD	List the event recording configuration for a stage in an Oracle Event Processing application.
STARTPLAYBACK	Starts playing back events for a stage in an Oracle Event Processing application.
STOPPLAYBACK	Stops playing back events for a stage in an Oracle Event Processing application.
CONFIGUREPLAYBACK	Configures the parameters for the event playback of a stage in an Oracle Event Processing application.
SCHEDULEPLAYBACK	Schedules the playback of events for a stage in an Oracle Event Processing application.
LISTPLAYBACK	List the event playback configuration for a stage in an Oracle Event Processing application.

A.13.1 STARTRECORD

Starts recording events for a particular stage of an Oracle Event Processing application.

Note:

Before you can use commands for controlling event record and playback on a stage, you must first configure the stage with the appropriate event record and playback options. See:

- CONFIGURERECORD
- *Developing Applications for Event Processing with Oracle Stream Explorer*
- *Schema Reference for Oracle Stream Explorer*.

If you configured the stage to start recording at a later time, that configuration is ignored and recording starts immediately.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
```

```
[ Common Arguments ]
STARTRECORD -application application -stage stage
```

Table A-20 STARTRECORD Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

Example

The following example shows how to start the recording of events on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle Event Processing instance:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     STARTRECORD -application helloworld -stage helloworldAdapter
```

Note:

Before you can use commands for controlling event record and playback on a stage, you must first configure the stage with the appropriate event record and playback options. See:

- CONFIGURERECORD
- *Developing Applications for Event Processing with Oracle Stream Explorer*
- *Schema Reference for Oracle Stream Explorer*.

A.13.2 STOPRECORD

Stops the recording of events for a stage of an Oracle Event Processing application in which the recording of events has been previously started.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
```

```
[ User Credentials Arguments ]
[ Common Arguments ]
STOPRECORD -application application -stage stage
```

Table A-21 STOPRECORD Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

Example

The following example shows how to stop the recording of events on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle Event Processing instance; it is assumed that the recording of events was previously started for the stage:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     STOPRECORD -application helloworld -stage helloworldAdapter
```

A.13.3 CONFIGURERECORD

Configures the parameters associated with the recording of events for a stage of an Oracle Event Processing application. Use this command to configure a stage for the first time for event recording or to change the data set name or provider name. For more information, see *Developing Applications for Event Processing with Oracle Stream Explorer*and *Schema Reference for Oracle Stream Explorer*.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    CONFIGURERECORD -application application -stage stage
        [-datasetName datasetname]
        [-storeProvider storeprovidername]
        [-eventTypes eventtypes]
        [-scheduleStartTime startime]
        [-scheduleEndTime endtime | -scheduleDuration duration]
```

Table A-22 CONFIGURERECORD Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage` *stage*	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.
`-datasetName` *datasetname*	Specifies the name of the dataset in which events are recorded.
`-storeProvider` *storeprovidername*	Specifies a valid `data-source` name defined in the Oracle Event Processing server `config.xml` file. To select the default BDB provider, leave this argument empty or specify an argument value of `default-provider`. For more information, see *Schema Reference for Oracle Stream Explorer*.
`-eventTypes` *eventtypes*	Specifies the comma-separated list of valid event type names to be recorded. Event types must be defined in the event type repository.

`-scheduleStartTime` *starttime*	Specifies the time when the recording should start.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should start on October 20, 2013, at 11:22:07 am, use the following value:
	`10-20-2013:11:22:07`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.

Argument	Definition

`-scheduleEndTime` *endtime*	Specifies the actual time when the recording should end.
	Express the end time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should end on October 20, 2013, at 6:00pm, use the following value:
	`10-20-2013:18:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	Specify `null` if you want the recording to run forever.
	You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

`-scheduleDuration` *duration*	Specifies the duration of time after which event recording for this stage ends. Specify `null` if you want the recording to run forever. The format is `HH:mm:ss`, such as `01:00:00`. You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

Example

The examples in this section show how to configure the recording of events of the `helloworldAdapter` of the `helloworld` application deployed to the specified Oracle Event Processing instance.

The following example specifies a start and end time for recording:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGURERECORD -application helloworld -stage helloworldAdapter
     -datasetName myds -storeProvider mysp
     -scheduleStartTime 10-20-2013:11:22:07 -scheduleEndTime 10-20-2013:18:00:00
```

The following example specifies a start time and a duration for recording:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGURERECORD -application helloworld -stage helloworldAdapter
     -datasetName myds -storeProvider mysp
     -scheduleStartTime 10-20-2013:11:22:07 -scheduleDuration 01:00:00
```

The following example specifies a start time and a duration of `null`, which means recording runs forever:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGURERECORD -application helloworld -stage helloworldAdapter
     -datasetName myds -storeProvider mysp
     -scheduleStartTime 10-20-2013:11:22:07 -scheduleDuration null
```

A.13.4 SCHEDULERECORD

Configures the schedule parameters associated with the recording of events for a stage of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    SCHEDULERECORD -application application -stage stage
        [-scheduleStartTime startime]
        [-scheduleEndTime endtime | -scheduleDuration duration]
```

Table A-23 SCHEDULERECORD Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage` *stage*	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

`-scheduleStartTime` *starttime*	Specifies the time when the recording should start.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should start on October 20,
	2013, at 11:22:07 am, use the following value:
	`10-20-2013:11:22:07`

Argument	Definition

`-scheduleEndTime` *endtime*	Specifies the actual time when the recording should end.
	Express the end time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should end on October 20,
	2013, at 6:00 pm, use the following value:
	`10-20-2013:18:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	Specify `null` if you want the recording to run forever.
	You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

`-scheduleDuration` *duration*	Specifies the duration of time after which event recording for this stage ends. Specify `null` if you want the recording to run forever. The format is `HH:mm:ss`, such as `01:00:00`. You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

Example

The examples in this section show how to configure the scheduling of recording of events of the `helloworldAdapter` of the `helloworld` application deployed to the specified Oracle Event Processing instance.

The following example specifies a start and end time for recording:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     SCHEDULERECORD -application helloworld -stage helloworldAdapter
     -scheduleStartTime 10-20-2013:11:22:07 -scheduleEndndTime 10-20-2013:18:00:00
```

A.13.5 LISTRECORD

Lists the event recording configuration for any particular stage of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    LISTRECORD -application application -stage stage
```

Table A-24 LISTRECORD Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

Example

The following example shows how to list the event recording configuration on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle Event Processing instance:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     LISTRECORD -application helloworld -stage helloworldAdapter
```

A.13.6 STARTPLAYBACK

Starts the playback of events of a particular stage of a Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    STARTPLAYBACK -application application -stage stage
        [-filterStartTime starttime]
        [-filterEndTime endtime | -filterDuration duration]       [-speed speed]        [-
repeat true | false]
```

Table A-25 STARTPLAYBACK Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

`-filterStartTime startime`	Specifies that only events with record-time greater than or equal to this value will be selected for playback.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`yyyy-mm-ddThh:mm:ss`
	For example, to play back only events with record-time greater than or equal to January 20, 2010, at 5:00am, use the following value:
	`2010-01-20T05:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	Specify `null` if you want to select all events for playback.

`-filterEndTime endtime`	Specifies only events with record-time less than or equal to this value will be selected for playback.
	Express the end time as an XMLSchema `dateTime` value of the form:
	`yyyy-mm-ddThh:mm:ss`
	For example, to play back only events with record-time less than or equal to January 20, 2010, at 6:00pm, use the following value:
	`2010-01-20T18:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	You can specify either `-filterEndTime` or `-filterDuration`, but not both.

Argument	Definition
`-filterDuration` *duration*	Specifies the filter applied to events in the event store. Only events that were recorded during the filter time will be selected for play back. Specify `null` if you want to select all events for playback. The format is `HH:mm:ss`, such as `01:00:00`. You can specify either `-filterEndTime` or `-filterDuration`, but not both.
`-speed` *speed*	Specifies the playback speed as a positive float. The default value is 1, which corresponds to normal speed. A value of 2 means that events will be played back 2 times faster than the original record speed. Similarly, a value of 0.5 means that events will be played back 2 times slower than the original record speed.
`-repeat` *repeat*	Specifies whether to playback events again after the playback of the specified time interval is over. Valid values are `true` or `false`. Default value is `false`. A value of `true` means that the repeat of playback continues an infinite number of times until it is deliberately stopped (see STOPPLAYBACK). A value of `false` means that events will be played back only once.

Example

The following example shows how to start the playback of events on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle Event Processing instance:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     STARTPLAYBACK -application helloworld -stage helloworldAdapter
```

A.13.7 STOPPLAYBACK

Stops the playback of events for a stage of an Oracle Event Processing application in which the playback of events has been previously started.

Syntax

Table A-26 STOPPLAYBACK Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    STOPPLAYBACK -application application -stage stage
```

Example

The following example shows how to stop the playback of events on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle Event Processing instance; it is assumed that the playback of events was previously started for the stage:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     STOPPLAYBACK -application helloworld -stage helloworldAdapter
```

A.13.8 CONFIGUREPLAYBACK

Configures the parameters associated with the playback of events for a stage of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    CONFIGUREPLAYBACK -application application -stage stage
      [-datasetName datasetname]
      [-storeProvider storeprovidername]
      [-eventTypes eventtypes]
      [-scheduleStartTime sstart]
      [-scheduleEndTime send | -scheduleDuration sduration]
      [-filterStartTime fstart] [-filterEndTime fend | -filterDuration fduration]
```

```
[-speed speed]
[-repeat true | false]
```

Table A-27 CONFIGUREPLAYBACK Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.
`-datasetName` *datasetname*	Specifies the name of the dataset in which events are recorded.
`-storeProvider` *storeprovidername*	Specifies a valid `data-source` name defined in the Oracle Event Processing server `config.xml` file. To select the default BDB provider, leave this argument empty or specify an argument value of `default-provider`. For more information, see *Schema Reference for Oracle Stream Explorer*.
`-eventTypes` *eventtypes*	Specifies the comma-separated list of valid event type names for playing back. Event types must be defined in the event type repository.

`-scheduleStartTime` *sstart*	Specifies the time when play back should start.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should start on October 20,
	2013, at 11:22:07 am, use the following value:
	`10-20-2013:11:22:07`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.

Argument	Definition

`-scheduleEndTime` *send*	Specifies the actual time when the play back should end.
	Express the end time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that play back should end on January 20, 2013, at 6:00pm, use the following value:
	`10-20-2013:18:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	Specify `null` if you want the recording to run forever.
	You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

`-scheduleDuration` *sduration*	Specifies the duration of time after which event playback for this stage ends. Specify `null` if you want the event playback to run forever. The format is `HH:mm:ss`, such as `01:00:00`. You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

`-filterStartTime` *fstart*	Specifies that only events with record-time greater than or equal to this value will be selected for playback.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`yyyy-mm-ddThh:mm:ss`
	For example, to play back only events with record-time greater than or equal to January 20, 2010, at 5:00am, use the following value:
	`2010-01-20T05:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	Specify `null` if you want to select all events for playback.

Argument	Definition

`-filterEndTime` *fend*	Specifies only events with record-time less than or equal to this value will be selected for playback.
	Express the end time as an XMLSchema `dateTime` value of the form:
	`yyyy-mm-ddThh:mm:ss`
	For example, to play back only events with record-time less than or equal to January 20, 2010, at 6:00pm, use the following value:
	`2010-01-20T18:00:00`
	For complete details of the XMLSchema `dateTime` format, see http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation.
	You can specify either `-filterEndTime` or `-filterDuration`, but not both.

`-filterDuration` *fduration*	Specifies the filter applied to events in the event store. Only events that were recorded during the filter time will be selected for play back. Specify `null` if you want to select all events for playback. The format is `HH:mm:ss`, such as `01:00:00`. You can specify either `-filterEndTime` or `-filterDuration`, but not both.
`-speed` *speed*	Specifies the playback speed as a positive float. The default value is 1, which corresponds to normal speed. A value of 2 means that events will be played back 2 times faster than the original record speed. Similarly, a value of 0.5 means that events will be played back 2 times slower than the original record speed.
`-repeat` *repeat*	Specifies whether to playback events again after the playback of the specified time interval is over. Valid values are `true` or `false`. Default value is `false`. A value of `true` means that the repeat of playback continues an infinite number of times until it is deliberately stopped (see STOPPLAYBACK). A value of `false` means that events will be played back only once.

Example

The examples in this section show how to configure the playback of events of the `helloworldAdapter` of the `helloworld` application deployed to the specified Oracle Event Processing instance.

The following example specifies a start and end time for playback and that the speed of playback should be twice the normal speed and that once the playback of events for the time interval is over, the playback should start again:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGUREPLAYBACK -application helloworld -stage helloworldAdapter
     -scheduleStartTime 10-20-2013:11:22:07 -scheduleEndTime 10-20-2013:18:00:00
     -speed 2 -repeat true
```

The following example specifies a start and a duration for playback, that the speed of playback is 2 times slower than normal, and that the playback of events should occur only once:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGUREPLAYBACK -application helloworld -stage helloworldAdapter
     -scheduleStartTime 10:20:2013:11:22:07 -scheduleEndTime 10-20-2013:18:00:00
     -speed 0.5 -repeat false
```

The following example specifies a start and a duration of `null`, which means playback will run forever at normal speed:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGUREPLAYBACK -application helloworld -stage helloworldAdapter
     -scheduleStartTime 10:20:2013:11:22:07 -scheduleDuration null
```

A.13.9 SCHEDULEPLAYBACK

Configures the schedule parameters associated with playing back of events for a stage of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    SCHEDULEPLAYBACK -application application -stage stage
        [-scheduleStartTime startime]
        [-scheduleEndTime endtime | -scheduleDuration duration]
```

Table A-28 SCHEDULEPLAYBACK Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

Argument	Definition

`-scheduleStartTime` *`starttime`*	Specifies the time when play back should start.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should start on October 20, 2013, at 11:22:07 am, use the following value:
	`10-20-2013:11:22:07`
	For complete details of the XMLSchema `dateTime` format, see `http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation`.
	For complete details of the XMLSchema `dateTime` format, see `http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation`.

`-scheduleEndTime` *`endtime`*	Specifies the actual time when the play back should end.
	Express the start time as an XMLSchema `dateTime` value of the form:
	`mm-dd-yyyy:hh:mm:ss`
	For example, to specify that recording should end on October 20, 2013, at 6:00 pm, use the following value:
	`10-20-2013:18:00:00`
	Specify `null` if you want the recording to run forever.
	You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

`-scheduleDuration` *`duration`*	Specifies the duration of time after which event playback for this stage ends. Specify `null` if you want the recording to run forever. The format is `HH:mm:ss`, such as `01:00:00`. You can specify either `-scheduleEndTime` or `-scheduleDuration`, but not both.

Example

The examples in this section show how to configure the schedule of playback of events of the `helloworldAdapter` of the `helloworld` application deployed to the specified Oracle Event Processing instance.

The following example specifies a start and end time for event playback:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     SCHEDULEPLAYBACK -application helloworld -stage helloworldAdapter
     -scheduleStartTime 10-20-2013:11:22:07 -scheduleEndndTime 10-20-2013:18:00:00
```

A.13.10 LISTPLAYBACK

Lists the event playback configuration for any particular stage of an Oracle Event Processing application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    LISTPLAYBACK -application application -stage stage
```

Table A-29 LISTPLAYBACK Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose event record and playback you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage` *stage*	Specifies the name of the particular stage, attached to the Oracle Event Processing application specified with the `-application` argument, whose event record and playback you want to manage.

Example

The following example shows how to list the event playback configuration on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle Event Processing instance:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     LISTPLAYBACK -application helloworld -stage helloworldAdapter
```

A.14 Commands for Monitoring Throughput and Latency

This section explains the commands for monitoring throughput and latency.

Table A-30 is an overview of commands for monitoring throughput and latency in an Oracle Event Processing application. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table A-30 Overview of Commands for Monitoring Throughput and Latency

Command	Description
MONITORAVGLATENCY	Monitors the *average amount of time it takes an event to pass through specified path of the EPN, or latency*.
MONITORMAXLATENCY	Monitors the *maximum amount of time it takes an event to pass through specified path of the EPN, or latency*.

Command	Description
MONITORAVGLATENCYTHRESHOLD	Monitors whether the average latency of events flowing through a path of the EPN crosses a specified threshold.
MONITORAVGTHROUGHPUT	Monitors the number of events flowing through the entry or exit points of a specified stage.

A.14.1 MONITORAVGLATENCY

Monitors the average amount of time, or *latency*, it takes an event to pass through a specified path of the EPN of the specified application.

You specify the start and end stages of the path, and whether it should start or end at the entry or exit points of each respective stage. If you specify the same stage for the start and end of the path, you can monitor the latency of events flowing through a single stage.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     MONITORAVGLATENCY -application application
     -startStage startStage -startStagePoint stagePoint
     -endStage endStage -endStagePoint stagePoint
     -avgInterval avgInterval -timeUnit timeUnit
```

Table A-31 MONITORAVGLATENCY Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose throughput and latency you want to monitor. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-startStage startStage`	Specifies the name of the stage that starts the path for which you want to monitor latency. The stage is in the application specified by the `-application` option.
`-startStagePoint startStagePoint`	Specifies the specific starting point for monitoring latency of the specified start stage. You can start monitoring from the entry or exit point of the start stage. Valid values are `entry` and `exit`. Default value is `entry`.

Argument	Definition
`-endStage endStage`	Specifies the name of the stage that ends the path for which you want to monitor latency. The stage is in the application specified by the `-application` option.
`-endStagePoint endStagePoint`	Specifies the specific ending point for monitoring latency of the specified end stage. You can end monitoring from the entry or exit point of the end stage. Valid values are `entry` and `exit`. Default value is `entry`.
`-avgInterval` *avgInterval*	Specifies the average interval across which average latency is calculated. Specify the units with the `-timeUnit` option; default is milliseconds. Default value is 100.

Example

The following example shows how to monitor the average latency of events flowing through the `cqlprocessor` component, from entry point to exit point, of the `helloworld` application. Note that because the same stage is specified for both the start and end stages (`cqlprocessor`), the latency monitoring is happening for just the events flowing through a single stage:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     MONITORAVGLATENCY -application helloworld -startStage cqlprocessor
     -startStagePoint entry -endStage cqlprocessor -endStagePoint exit
     -avgInterval 100 -timeUnit MILLISECONDS
```

A.14.2 MONITORAVGLATENCYTHRESHOLD

Specifies whether the average latency of events between the start- and end-points of a path crosses a specified threshold.

You specify the start and end stages of the path, and whether it should start or end at the entry or exit points of each respective stage. If you specify the same stage for the start and end of the path, you can monitor the latency threshold of events flowing through a single stage.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    MONITORAVGLATENCYTHRESHOLD -application application
    -startStage startStage -startStagePoint stagePoint
    -endStage endStage -endStagePoint stagePoint
    -avgInterval avgInterval -timeUnit timeUnit -threshold threshold
```

Table A-32 MONITORAVGLATENCYTHRESHOLD Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle Event Processing application whose throughput and latency you want to monitor. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-startStage startStage`	Specifies the name of the stage that starts the path for which you want to monitor the latency threshold. The stage is in the application specified by the `-application` option.
`-startStagePoint startStagePoint`	Specifies the specific starting point for monitoring the latency threshold of the specified start stage. You can start monitoring from the entry or exit point of the start stage. Valid values are `entry` and `exit`. Default value is `entry`.
`-endStage endStage`	Specifies the name of the stage that ends the path for which you want to monitor the latency threshold. The stage is in the application specified by the `-application` option.
`-endStagePoint endStagePoint`	Specifies the specific ending point for monitoring the latency threshold of the specified end stage. You can end monitoring from the entry or exit point of the end stage. Valid values are `entry` and `exit`. Default value is `entry`.
`-avgInterval` *avgInterval*	Specifies the average interval across which average the latency threshold is calculated. Default value is 100. Specify the units with the `-timeUnit` option; default is milliseconds.
`-threshold` *threshold*	Specifies the threshold value above which the metric event will be outputted at the end of every average interval. Default is 100. Specify the units with the `-timeUnit` option; default is milliseconds.

Example

The following example shows how to monitor the average latency threshold of events above 10 seconds average latency on the `cqlprocessor` stage, from entry point to exit point, of the `helloworld` application.

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
```

```
-username wlevs -password wlevs
MONITORAVGLATENCY -application helloworld -startStage cqlprocessor
-startStagePoint entry -endStage cqlprocessor -endStagePoint exit
-avgInterval 100 -timeUnit MILLISECONDS -threshold 100
```

A.14.3 MONITORMAXLATENCY

Monitors the maximum latency of events flowing through a specified path of the EPN of the specified application.

You specify the start and end stages of the path, and whether it should start or end at the entry or exit points of each respective stage. If you specify the same stage for the start and end of the path, you can monitor the maximum latency of events flowing through a single stage.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    MONITORMAXLATENCY -application application
    -startStage startStage -startStagePoint stagePoint
    -endStage endStage -endStagePoint stagePoint
    -maxInterval maxInterval -timeUnit timeUnit
```

Table A-33 MONITORMAXLATENCY Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose throughput and latency you want to monitor. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-startStage startStage`	Specifies the name of the stage that starts the path for which you want to monitor the maximum latency. The stage is in the application specified by the `-application` option.
`-startStagePoint startStagePoint`	Specifies the specific starting point for monitoring the maximum latency of the specified start stage. You can start monitoring from the entry or exit point of the start stage. Valid values are `entry` and `exit`. Default value is `entry`.

Argument	Definition
`-endStage endStage`	Specifies the name of the stage that ends the path for which you want to monitor the maximum latency. The stage is in the application specified by the `-application` option.
`-endStagePoint endStagePoint`	Specifies the specific ending point for monitoring the maximum latency of the specified end stage. You can end monitoring from the entry or exit point of the end stage. Valid values are `entry` and `exit`. Default value is `entry`.
`-maxInterval` *`maxInterval`*	Specifies the interval across which maximum latency is calculate. Default value is 100. Specify the units with the `-timeUnit` option; default is milliseconds.

Example

The following example shows how to monitor the maximum latency of events flowing through the `cqlprocessor` stage, from entry point to exit point, of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     MONITORMAXLATENCY -application helloworld -startStage cqlprocessor
     -startStagePoint entry -endStage cqlprocessor -endStagePoint exit
     -maxInterval 100 -timeUnit MILLISECONDS
```

A.14.4 MONITORAVGTHROUGHPUT

Monitors the average number of events flowing through the entry or exit point of a stage of the EPN of the specified application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    MONITORAVGTHROUGHPUT -application application
    -stage stage -StagePoint stagePoint
    -throughputInterval throughputInterval -avgInterval avgInterval
    -timeUnit timeUnit
```

Table A-34 MONITORAVGLATENCY Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose throughput and latency you want to monitor. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-stage stage`	Specifies the name of the stage for which you want to monitor throughput of events. The stage is in the application specified by the `-application` option.
`-stagePoint stagePoint`	Specifies whether you want to monitor throughput at the entry- or exit- point of the specified stage. Valid values are `entry` and `exit`. Default value is `entry`.
`-throughputInterval throughputInterval`	Specifies the throughput interval across which throughput is calculated. Default value is 100. Specify the units with the `-timeUnit` option; default is milliseconds.
`-avgInterval avgInterval`	Specifies the average interval across which average throughput is calculated. Default value is 100. Specify the units with the `-timeUnit` option; default is milliseconds.

Example

The following example shows how to monitor the number of events flowing through the entry point of the `cqlprocessor` stage of the `helloworld` application:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     MONITORMAXLATENCY -application helloworld
     -stage cqlprocessor -stagePoint entry
     -throughputInterval 100 -avgInterval 100 -timeUnit MILLISECONDS
```

A.15 Commands for Managing Configuration History

This section explains the commands for managing configuration history.

Table A-35 is an overview of commands that manage the configuration history of Oracle Event Processing components. For more information, see Manage Configuration History.

Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table A-35 Overview of Commands for Managing Configuration History

Command	Description
CONFIGHISTORY	Returns the list of configuration history management commands.
DELETECONFIGCHANGEHISTORY	Removes change records for a specified time period.
LISTCHANGERECORDS	Returns a list of the change records of an application.
LISTRESOURCEREVISIONS	Returns a list of the configuration resource revisions of an application.
UNDOCONFIGCHANGE	Rolls back a change record specified by change record ID.

A.15.1 CONFIGHISTORY

This section explains the CONFIGHISTORY command.

Returns the list of configuration history management commands.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    LISTCHANGERECORDS
    LISTRESOURCEREVISIONS     UNDOCONFIGCHANGE
    DELETECONFIGCHANGEHISTORY
```

Example

The following example shows how to list the configuration history management commands:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     LISTCHANGERECORDS
```

A.15.2 DELETECONFIGCHANGEHISTORY

Returns the list of configuration history management commands.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    DELETECONFIGCHANGEHISTORY -application application -startTime starttime -endTime
endtime
```

Table A-36 DELETECONFIGCHANGEHISTORY Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose Oracle CQL rules you want to manage. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-startTime starttime`	Specifies the beginning of the time period to delete change records. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.
`-endTime end-ime`	Specifies the end of the time period to delete change records. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.

Example

The following example shows how to list the configuration history management commands:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     CONFIGHISTORY
```

A.15.3 LISTCHANGERECORDS

Returns a list of the change records of an application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    LISTCHANGERECORDS -application application -startTime starttime -endTime endtime
```

Table A-37 GETRULE Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose change records you want to browse. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-startTime starttime`	Specifies the beginning of the time period to filter the display of change records. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.
`-endTime end-ime`	Specifies the end of the time period to filter the display of change records. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.

Example

The following example shows how to list all the change records created between 11:10:07 and 11:22:07 on 20 November 2007 for the application `helloworld`:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     LISTCHANGERECORDS -application helloworld -startTime 10-20-2007:11:10:07
     -endTime 10-20-2007:11:22:07
```

A.15.4 LISTRESOURCEREVISIONS

Returns a list of the configuration resource revisions of an application.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    LISTRESOURCEREVISIONS -application application -startTime starttime -endTime endtime
```

Table A-38 GETRULE Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose resource revisions you want to browse. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see Oracle Event Processing *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-startTime starttime`	Specifies the beginning of the time period to filter the list of resource revisions. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.
`-endTime end-ime`	Specifies the end of the time period to filter the list of resource revisions. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.

Example

The following example shows how to list all the resource revisions created between 11:10:07 and 11:22:07 on 20 November 2007 for the application `helloworld`:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
      LISTRESOURCEREVISIONS -application helloworld -startTime 10-20-2007:11:10:07
     -endTime 10-20-2007:11:22:07
```

A.15.5 UNDOCONFIGCHANGE

Rolls back the changes defined by the change record specified by change record ID.

Syntax

```
java wlevs.Admin
    [ Connection Arguments ]
    [ User Credentials Arguments ]
    [ Common Arguments ]
    UNDOCONFIGCHANGE -application application -recordId changerecordid
```

Table A-39 GETRULE Arguments

Argument	Definition
`-application application`	Specifies the name of the Oracle Event Processing application whose change records you want to undo. To get the exact name of your application, you can: • Use `wlevs.Admin` to query for the name (see Query for Application and Processor Names). • Use the Oracle Event Processing Visualizer: Start the Oracle Event Processing Visualizer (see *Using Visualizer for Oracle Stream Explorer*). In the left pane, navigate to and expand the **Applications** node of the Oracle Event Processing instance to which the application is deployed. Each node under the Applications node is named with the exact application name. • Look at the `MANIFEST.MF` file of the application; the application name is specified by the `Bundle-SymbolicName` header.
`-recordId changerecordid`	Specifies the identifier of the change record to undo. To get the change record identifier, you can use: • LISTCHANGERECORDS • LISTRESOURCEREVISIONS.

Example

The following example shows how to roll back all the resource revisions created between 11:10:07 and 11:22:07 on 20 November 2007 for the application `helloworld`:

```
java wlevs.Admin
     -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
     -username wlevs -password wlevs
     UNDOCONFIGCHANGE -application helloworld -recordId tr.1267607521409.10110
```

B

Deployer Command-Line Reference

This appendix provides a reference to the Oracle Event Processing `Deployer` utility, which you can use to deploy Oracle Event Processing applications.

This appendix includes the following sections:

- Overview of Using the Deployer Utility
- Configure the Deployer Utility Environment
- Run the Deployer Utility Remotely
- Syntax to Invoke the Deployer Utility
- Deployer Utility Examples.

B.1 Overview of Using the Deployer Utility

The `Deployer` utility is a Java-based deployment utility that provides administrators and developers command-line based operations for deploying Oracle Event Processing applications. In the context of Oracle Event Processing deployment, an application is defined as an OSGi bundle at http://www.osgi.org/ JAR file that contains the following artifacts:

- The compiled Java class files that implement some of the components of the application, such as the adapters, adapter factory, and POJO that contains the business logic.
- One or more Oracle Event Processing configuration XML files that configure the components of the application, such as the processor, adapter, or streams.

  The configuration files must be located in the `META-INF/wlevs` directory of the OSGi bundle JAR file.
- An EPN assembly file that describes all the components of the application and how they are connected to each other. The EPN assembly file extends the standard Spring context file.

  The EPN assembly file must be located in the `META-INF/spring` directory of the OSGi bundle JAR file.
- A `MANIFEST.MF` file that describes the contents of the JAR.

See *Developing Applications for Event Processing with Oracle Stream Explorer* for information about creating this deployment bundle.

The Deployer utility uses HTTP to connect to Oracle Event Processing, which means that you must configure Jetty for the server instance to which you are deploying your application. See *Administering Oracle Stream Explorer*.

Oracle Event Processing uses the `deployments.xml` file to internally maintain its list of deployed application OSGi bundles. This file is located in the *`DOMAIN_DIR/servername`* directory, where *`DOMAIN_DIR`* refers to the main domain directory corresponding to the server instance to which you are deploying your application and *`servername`* refers to the server instance itself.

Caution:

The XSD for the `deployments.xml` file is provided for your information only. Oracle does not recommend updating the `deployments.xml` file manually.

B.2 Configure the Deployer Utility Environment

Before you can use the Deployer utility, you must configure your environment appropriately.

Configure the Deployer utility environment:

Perform the following steps to configure the deployer utility environment:

1. Install and configure the Oracle Event Processing software.
2. Update your `CLASSPATH` variable to include the `wlevsdeploy.jar JAR` file, located in the `/Oracle/Middleware/my_oep/oep/bin` directory.

B.3 Run the Deployer Utility Remotely

Sometimes it is useful to run the `Deployer` utility on a computer different from the computer on which Oracle Event Processing is installed and running.

Run the Deployer utility remotely:

1. Copy the following JAR files from the computer on which Oracle Event Processing is installed to the computer on which you want to run the Deployer utility; you can copy the JAR files to the directory name of your choice:
   - */Oracle/Middleware/my_oep/oep/bin/*wlevsdeploy.jar
2. Set your `CLASSPATH` in one of the following ways:
   - Implicitly set your `CLASSPATH` by using the `-jar` argument when you run the utility; set the argument to the *`NEW_DIRECTORY`*`/wlevsdeploy.jar` file, where *`NEW_DIRECTORY`* refers to the directory on the remote computer into which you copied the required JAR file. When you use the `-jar` argument, you do not specify the `Deployer` utility name at the command line.
   - Explicitly update your `CLASSPATH` by adding the JAR file you copied to the remote computer to your `CLASSPATH` environment variable:
3. Invoke the `Deployer` utility as described in the next section.

B.4 Syntax to Invoke the Deployer Utility

The syntax for using the Deployer utility is as follows:

```
java -jar wlevsdeploy.jar
    [Connection Arguments]
    [User Credential Arguments]
    [Deployment Commands]
```

The following sections describe the various arguments and commands you can use with the Deployer utility. See Deployer Utility Examples for specific examples of using the utility.

B.4.1 Connection Arguments

Table B-1 lists the connection arguments you can specify with the Deployer utility.

Table B-1 Connection Arguments

Argument	Description

`-url url`	Specifies the URL to the Deployer utility for the Oracle Event Processing instance to which you want to deploy the OSGI bundle.
	The URL takes the following form:
	`http://host:port/wlevsdeployer`
	where:
	• *`host`* refers to the host name of the computer on which Oracle Event Processing is running.
	• *`port`* refers to the port number to which Oracle Event Processing listens. Its value is `9002` by default. This port is specified in the `config.xml` file that describes your Oracle Event Processing domain. By default it is located in the *`Oracle/Middleware/my_oep/user_projects/domains/<domainname>/ <servername>`*`/config` directory.
	The port number is the value of the `<Port>` child element of the `<Netio>` element:
	`<Netio>`
	`  <Name>NetIO</Name>`
	`  <Port>9002</Port>`
	`</Netio>`
	If you configure the Oracle Event Processing server for SSL-only connections (HTTPS-Only Connections), use the value of the `<Port>` child element of the SSL `<Netio>` element:
	`<Netio>`
	`  <name>sslNetIo</name>`
	`  <port>9003</port>`
	`  <ssl-config-bean-name>sslConfig</ssl-config-bean-name>`
	`</Netio>`
	For example, if Oracle Event Processing is running on host `ariel` at port `9002`, then the URL would be:
	`http://ariel:9002/wlevsdeployer`

B.4.2 User Credential Arguments

Table B-2 lists the user credential arguments you can specify with the Deployer utility.

Table B-2 User Credential Arguments

Argument	Description
`-user` *`username`*	User name of the Oracle Event Processing administrator. If you supply the `-user` option but you do not supply a corresponding `-password` option, the Deployer utility prompts you for the password.
`-password` *`password`*	Password of the Oracle Event Processing administrator. **Note:** This argument is deprecated and may be removed in a later release. Oracle recommends that you do not use this argument.

B.4.3 Deployment Commands

Table B-3 lists the deployment commands you can specify with the Deployer utility.

Table B-3 Deployment Commands

Command	Description
`-encrypt`	Encrypts the user name and password and writes it to output file.
`-encryptoutfile` *`encryptoutfile`*	Specifies that *encryptoutfile* should be used to write encrypted the user and password.

`-group` *`groupname`*	Specifies that the deploy command (install, uninstall, update, suspend, or resume) applies to a target group, or more specifically, to the set of running servers within that group.
	To specify the domain group, use the keyword `all`, such as:
	`-group all`
	To specify a custom group, simply specify the name of the group:
	`-group my_group`
	Note: You may only deploy to a group if the server is part of a multiserver domain (that is, if clustering is enabled). You may not deploy to a group if the server is part of a standalone-server domain (that is, if clustering is disabled). For more information, see About Multiserver Domains.

`-help`	Prints a message describe command syntax and arguments.

Command	Description
`-install` *`bundle`*	Installs the specified OSGi bundle to the specified Oracle Event Processing instance. The *`bundle`* parameter refers to a filename that is local to the computer from which you execute the `Deployer` utility. Be sure to specify the full pathname of the bundle if it is not located relative to the directory from which you are running the Deployer utility. In particular, Oracle Event Processing: • Copies the specified bundle to the domain directory. • Searches the `META-INF/wlevs` directory in the bundle for the component configuration files and extracts them to the domain directory. • Updates the internal deployment registry. • Starts the application. The incoming adapters immediately start receiving data.
`-resume` *`name`*	Resumes a previously suspended OSGI bundle on the specified Oracle Event Processing instance; the configured adapters once again start immediately receiving incoming data. The *`name`* parameter refers to the symbolic name of the OSGi bundle that you want to stop. The symbolic name is the value of the `Bundle-SymbolicName` header in the bundle's `MANIFEST.MF` file: `Bundle-SymbolicName: myApp`
`-status` *`name`*	Returns status information about a currently installed OSGi bundle. The *`name`* parameter refers to the symbolic name of the OSGi bundle for which you want status information. The symbolic name is the value of the `Bundle-SymbolicName` header in the bundle's `MANIFEST.MF` file: `Bundle-SymbolicName: myApp`
`-suspend` *`name`*	Suspends a currently running OSGI bundle which was previously installed to the specified Oracle Event Processing instance. The *`name`* parameter refers to the symbolic name of the OSGi bundle that you want to start. The symbolic name is the value of the `Bundle-SymbolicName` header in the bundle's `MANIFEST.MF` file: `Bundle-SymbolicName: myApp`
`-uninstall` *`name`*	Removes the existing bundle from the specified Oracle Event Processing instance. The *`name`* parameter refers to the symbolic name of the OSGi bundle that you want to remove. The symbolic name is the value of the `Bundle-SymbolicName` header in the bundle's `MANIFEST.MF` file: `Bundle-SymbolicName: myApp` In particular, Oracle Event Processing: • Removes the specified OSGi bundle from the domain directory. • Removes the bundles from the internal deployment registry.

Command	Description
`-update bundle`	Updates the existing OSGi bundle with new application code. The `bundle` parameter refers to a filename that is local to the computer from which you execute the `Deployer` utility. Be sure to specify the full pathname of the bundle if it is not located relative to the directory from which you are running the Deployer utility. In particular, Oracle Event Processing: • Copies the updated bundles to the domain directory. • Searches the `META-INF/wlevs` directory in the updated bundle for the updated component configuration files and extracts them to the domain directory. • Updates the internal deployment registry with the updated information.
`-userconfigfile userconfigfile`	Specifies that *userconfigfile* (security-config.xml) should be used to retrieve encrypted user name and password from the file.
`-userkeyfile userkeyfile`	Specifies that *userkeyfile* (.msainternal.dat) should be used to get the encryption key used to encrypt the password in the user `config` file.

B.5 Deployer Utility Examples

The following examples show how to use the Deployer utility. In all the examples, Oracle Event Processing is running on host `ariel`, listening at port `9002`, and the user name/password of the server administrator is `oepadmin/welcome1`, respectively. For clarity, the examples are shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

```
java -jar wlevsdeploy.jar
     -url http://ariel:9002/wlevsdeployer -user wlevs -password wlevs
     -install /application/bundles/com.my.exampleApp_1.0.0.0.jar
```

The preceding example shows how to install an OSGi bundle called `com.my.exampleApp_1.0.0.0.jar`, located in the `/application/bundles` directory.

The next command shows how to resume this application after it has been suspended:

```
java com.bea.wlevs.deployment.Deployer
     -url http://ariel:9002/wlevsdeployer -user wlevs -password wlevs
     -resume exampleApp
```

The next example shows how to uninstall the application, which removes all traces of it from the domain directory:

```
java com.bea.wlevs.deployment.Deployer
     -url http://ariel:9002/wlevsdeployer -user wlevs -password wlevs
     -uninstall exampleApp
```

The following example shows how to install an application called `strategies_1.0.jar` to the `strategygroup`; this example also shows how to use the `-jar` command of the `java` utility:

```
java -jar wlevsdeploy.jar
     -url http://ariel:9002/wlevsdeployer -install strategies_1.0.jar
     -group strategygroup
```

C

Security Utilities Command-Line Reference

This appendix provides a reference to the Oracle Event Processing security utilities, including `cssconfig`, `encryptMSAConfig`, and `GrabCert`, which are all utilities for generating security configuration files, encrypting cleartext passwords, and generating a trust keystore.

This appendix includes the following sections:

- The cssconfig Command-Line Utility
- The encryptMSAConfig Command-Line Utility
- The GrabCert Command-Line Utility
- The passhash Command-Line Utility
- The policygen Command-Line Utility
- The encrypttool Command-Line Utility.

Except where otherwise noted, the commands are located in `/Oracle/Middleware/my_oep/oep/bin`.

Note:

The `GrabSert`, `passgen`, and `secgen` command-line utilities are deprecated. Configuration Wizard and Oracle Event Processing Visualizer perform the `passgen` and `secgen` tasks for you.

C.1 The cssconfig Command-Line Utility

Use the `cssconfig` command-line utility to generate a security configuration file (`security.xml`) that uses a password policy.

- `cssconfig.cmd` (Windows)
- `cssconfig.sh` (UNIX)

The Unix version of this utility starts with the `#!/bin/ksh` directive. On most Unix systems, this forces the Korn Shell program to be used when using the utility. If the `ksh` program is not present in the `bin` directory or if the shell language used cannot properly execute the utility, run the utility as shown below:

```
$PATH_TO_KSH_BIN/ksh -c cssconfig.sh
```

where *`PATH_TO_KSH_BIN`* is the fully qualified path to the `ksh` program.

Syntax

```
cssconfig -p propertyfile [-c configfile] -i inputkeyfile [-d]
```

Table C-1 encryptMSAConfig Arguments

Option	Description	Default Value
propertyfile	Required. A file that contains security configuration properties provided by the user to define the required configuration. Configure SSL Manually.	
configfile	Optional. The name of the generated file. This property is optional.	security.xml
inputkeyfile	The fully qualified name of the input key file used to generate the security configuration file. Set this option to the security-key.dat file in the config directory.	
-d	Use the -d option to enable debugging.	

C.2 The encryptMSAConfig Command-Line Utility

This tool is not available on Oracle WebLogic Server. Use the encryptMSAConfig encryption command-line utility to encrypt cleartext passwords.You can use encryptMSAConfig to encrypt the server config.xml and security.xml files, and the application configuration credential.

- encryptMSAConfig.cmd (Windows)
- encryptMSAConfig.sh (UNIX)

Cleartext passwords are specified by the <password> element, in XML files. Examples of XML files that can contain the <password> elements include:

- config.xml
- security-config.xml
- Component configuration files

Syntax

```
encryptMSAConfig directory XML_file aesinternal.dat_file
```

Table C-2 encryptMSAConfig Arguments

Option	Description
directory	The name of the directory that contains the XML file with the cleartext <password /> element.
XML_file	The name of the XML file.

Option	Description
aesinternal.dat_file	The location of the `.aesinternal.dat` key file associated with your domain. The key file encrypts the `<password />` element in the XMLfile parameter. The `aesinternal.dat_file` file is located in the /Oracle/Middleware/my_oep/`user_projects/domains/`*SERVER* directory
`-noinput`	Use the `-noinput` option to instruct `GrabCert` to copy all certificates from *host*. Omit the `-noinput` option to instruct `GrabCert` to list all available certificates from *host* and prompt you to select one.

For example:

```
pwd C:\Oracle\Middleware\my_oep\user_projects\domains\oep_domain\defaultserver

C:\Oracle\Middleware\my_oep\oep\bin\encryptMSAConfig.cmd . config\config.xml
  .aesinternal.dat
```

After you run the command, the value of the `password` element in *XML_file* is encrypted.

C.3 The GrabCert Command-Line Utility

Use the `GrabCert` command-line utility to generate a trust keystore that includes the certificate from an existing trust keystore.

The `GrabCert` utility is located in the */Oracle/Middleware/my_oep/oep/utils/security/wlevsgrabcert.jar* file.

Syntax

```
java GrabCert host:secureport [-alias=alias] [-noinput] [truststorepath]
```

Table C-3 GrabCert Arguments

Option	Description	Default Value
host	The host name of the Oracle Event Processing server from which to copy the certificate.	
secureport	The SSL port on *host*. For more information, see Configure SSL Manually.	9003
alias	The alias for the certificate in the trust keystore.	*host*
`-noinput`	Use the `-noinput` option to instruct `GrabCert` to copy all certificates from *host*. Omit the `-noinput` option to instruct `GrabCert` to list all available certificates from *host* and prompt you to select one.	
truststorepath	The full pathname of the generated trust keystore file on *host*.	`evstrust.jks`

Examples

For example:

```
java GrabCert ariel:9003 -alias=ariel evstrust.jks
```

For other examples, see Configure SSL in a Multiserver Domain for Visualizer.

C.4 The passhash Command-Line Utility

This tool is not available on Oracle WebLogic Server. Use the `passhash` command-line utility to encrypt a password to use in the `atnstore.txt` file.

- passgen.cmd (Windows)
- passgen.sh (UNIX)

Note:

To get command-line help for this tool, use `-help` instead of `-h`.

Syntax

```
passhash [pasword]
```

The `password` parameter is a plain text string. The command output is a hashed encrypted string using the MD5/SHA encryption algorithm.

```
./passhash.sh
Password ("quit" to end): 4444
{SHA-1}+wQ3QDREP82FCrpDYspXM8SAlaMCx0o=
Password ("quit" to end): quit
```

C.5 The policygen Command-Line Utility

Use the `policygen` command-line utility to convert an entitlement file to an XACML LDIFT file or to an XACML file.

Syntax

```
policygen [-h]
policygen [-s] [-l] | -s] [-x] [entitlementInputFile] [xacmlOutputFile]
```

Table C-4 policygen Arguments

Option	Description
`-h`	Print command help to the console.
`-s`	Generate a standard XACML policy inside an XACML LDIFT file or in an XACML file. When no -l or -s is specified, an XACML LDIFT file is generated. When no -s option is specified, an XACML policy file is generated.
`-l`	`Generate an XACML LDIFT file.`

Option	Description
`-x`	Generate an XACML policy file.
entitlementInputFile	The name and location of the input entitlement XML file.
xacmlOutputFile	The name and location of the output XACML file.

Examples

The following example generates an XACML policy file:

```
./policygen.sh -l entitlementinputfile.xml xacmloutputfile.xml
```

C.6 The encrypttool Command-Line Utility

Use the `encrypttool` command-line utility to encrypt and decrypt files. This command uses an `EncryptedStreamFactory` object for encryption and decryption. The encryption result is a binary encrypted file. All content in the input file is encrypted using the AES/DES encryption algorithm.

Syntax

```
encrypttool [-h]
encrypttool [-encrypt] [-decrypt] [-password password] [-algorithm algorithm]
[inputfilename] [outputfilename]
```

Table C-5 encrypttool Arguments

Option	Description
`-encrypt`	Encrypt the input file and save the encryption results to the encrypted output file.
`-decrypt`	Decrypt the input file and save the decryption results to the unencrypted output file.
`-password`	The password that is required to encrypt or decrypt a file. If you do not provide the password, the system prompts you for it.
`-algorithm`	The encryption or decryption algorithm to use for the operation. The legal values are AES and DES. DES is the default.
inputfilename	The location and name of the input file to be encrypted or decrypted.
outputfilename	The name and location of the output file in which to save the encryption or decryption results. If you do not specify an output file, the results are printed to the console.

Examples

The following example uses the `mypassword` password to encrypt the `textToEncrypt` file with the AES encryption algorithm and saves the results to the `encryptedText` file.

```
encrypttool -encrypt -password mypassword -algorithm AES textToEncrypt encryptedText
```