Oracle® Complex Event Processing

Administration and Configuration Guide

Release 10*g*R3 (10.3)

September 2008

ORACLE

Oracle Complex Event Processing Administration and Configuration Guide, Release 10*g*R3 (10.3)

Contents

3. Configuring Oracle CEP Servers

4. Configuring and Using Oracle CEP Multi-Server Domains

5. wlevs.Admin Command-Line Reference

6. Managing Applications, Servers, and Domains Using MBeans

7. Configuring Security for Oracle CEP

8. Configuring Jetty for Oracle Complex Event Processing

9. Configuring JMX for Oracle Complex Event Processing

10. Configuring Access to a Relational Database

11. Configuring the HTTP Publish-Subscribe Server

12. Configuring Logging and Debugging

CHAPTER 1

Introduction and Roadmap

This section describes the contents and organization of this guide—*Oracle Complex Event Processing Administration and Configuration Guide*.

Note: In this section, *Oracle Complex Event Processing* is also referred to as *Oracle CEP*, for simplicity.

Document Scope and Audience

This document is a resource for software developers who develop event driven real-time applications. It also contains information that is useful for business analysts and system architects who are evaluating Oracle CEP or considering the use of Oracle CEP for a particular application.

The topics in this document are relevant during the design, development, configuration, deployment, and performance tuning phases of event driven applications. The document also includes topics that are useful in solving application problems that are discovered during test and pre-production phases of a project.

It is assumed that the reader is familiar with the Java programming language and Spring.

Oracle CEP Documentation Set

This document is part of a larger Oracle CEP documentation set that covers a comprehensive list of topics. The full documentation set includes the following documents:

- *Oracle CEP Getting Started*
- *Oracle CEP Application Development Guide*
- *Oracle CEP Administration and Configuration Guide*
- *Oracle CEP EPL Reference Guide*
- *Oracle CEP Reference Guide*
- *Oracle CEP Release Notes*
- *Oracle CEP Visualizer Help*
- *Oracle CEP Type 4 JDBC Drivers*

See the main Oracle CEP documentation page for further details.

Guide to This Document

This document is organized as follows:

- This chapter, Chapter 1, "Introduction and Roadmap," introduces the organization of this guide and the Oracle CEP documentation set and samples.
- Chapter 2, "Creating and Updating an Oracle CEP Standalone-Server Domain," describes how to create a new Oracle CEP domain.
- Chapter 3, "Configuring Oracle CEP Servers," describes how to start and stop an Oracle CEP server, as well as how to configure it.
- Chapter 4, "Configuring and Using Oracle CEP Multi-Server Domains," describes how to cluster servers in a domain together and deploy applications to it.
- Chapter 5, "wlevs.Admin Command-Line Reference," provides reference information about using the `wlevs.Admin` utility to configure Oracle CEP and the EPL rules attached to a particular application.
- Chapter 7, "Configuring Security for Oracle CEP," provides information about configuring various types of security for Oracle CEP.

- Chapter 8, "Configuring Jetty for Oracle Complex Event Processing," describes how to configure some features of the microServices Architecture, in particular the Jetty Web Service, work managers, and net IO.
- Chapter 9, "Configuring JMX for Oracle Complex Event Processing," describes how to configure some features of the microServices Architecture, in particular RMI, JNDI, and JMX.
- Chapter 10, "Configuring Access to a Relational Database," describes how to configure access to a relational database.
- Chapter 11, "Configuring the HTTP Publish-Subscribe Server," describes how to configure and use the HTTP publish-subscribe server, one of the features of Oracle CEP.
- Chapter 12, "Configuring Logging and Debugging," describes how to configure the logging and debugging features of the microServices Architecture.

Samples for the Oracle CEP Application Developer

In addition to this document, Oracle provides a variety of code samples for Oracle CEP application developers. The examples illustrate Oracle CEP in action, and provide practical instructions on how to perform key development tasks.

Oracle recommends that you run some or all of the examples before programming and configuring your own event driven application.

Note: When you initially install Oracle CEP, you must chose the `Custom` option to also install the examples. The `Typical` option does *not* include the examples.

If you previously installed Oracle CEP using the `Typical` option, and you now want to also install the examples, re-run the Oracle CEP installation process and specify the same Oracle CEP home directory; a later step in the installation process allows you to then install just the examples.

The examples are distributed in two ways:

- Pre-packaged and compiled in their own domain so you can immediately run them after you install the product.
- Separately in a Java source directory so you can see a typical development environment setup.

The following four examples are provided in both their own domain and as Java source in this release of Oracle CEP:

- HelloWorld—Example that shows the basic elements of an Oracle CEP application. See Hello World Example for additional information.

  The HelloWorld domain is located in *ORACLE_CEP_HOME*\ocep_10.3\samples\domains\helloworld_domain, where *ORACLE_CEP_HOME* refers to the Oracle CEP installation directory, such as c:\oracle_cep.

  The HelloWorld Java source code and configuration files are located in *ORACLE_CEP_HOME*\ocep_10.3\samples\source\applications\helloworld.

- ForeignExchange (FX)—Example that includes multiple adapters, streams, and complex event processor with a variety of EPL rules, all packaged in the same Oracle CEP application. See Foreign Exchange (FX) Example for additional information.

  The ForeignExchange domain is located in *ORACLE_CEP_HOME*\ocep_10.3\samples\domains\fx_domain, where *ORACLE_CEP_HOME* refers to the Oracle CEP installation directory, such as c:\oracle_cep.

  The ForeignExchange Java source code and configuration files are located in *ORACLE_CEP_HOME*\ocep_10.3\samples\source\applications\fx.

- Signal Generation—Example that receives simulated market data and verifies if the price of a security has fluctuated more than two percent, and then detects if there is a *trend* occurring by keeping track of successive stock prices for a particular symbol.See Signal Generation Example for additional information.

  The Signal Generation domain is located in *ORACLE_CEP_HOME*\ocep_10.3\samples\domains\signalgeneration_domain, where *ORACLE_CEP_HOME* refers to the Oracle CEP installation directory, such as c:\oracle_cep.

  The Signal Generation Java source code and configuration files are located in *ORACLE_CEP_HOME*\ocep_10.3\samples\source\applications\signalgeneration.

- Record and Playback—Example that shows how to configure the recording and playback of events to a persistent event store, as well as how to use the built-in HTTP pub-sub adapter to publish messages to a channel. See Event Record and Playback Example for additional information.

  The Record and Playback domain is located in *ORACLE_CEP_HOME*\ocep_10.3\samples\domains\recplay_domain, where *ORACLE_CEP_HOME* refers to the Oracle CEP installation directory, such as c:\oracle_cep.

The Record and Playback Java source code and configuration files are located in *ORACLE_CEP_HOME*\ocep_10.3\samples\source\applications\recplay.

CHAPTER 2

Creating and Updating an Oracle CEP Standalone-Server Domain

This section contains information on the following subjects:

Overview of Oracle CEP Servers and Domains

An Oracle CEP *server* consists of logically related resources and services to which you deploy Oracle CEP applications. Services include the Jetty (HTTP server), JDBC datasources, the HTTP publish-subscribe server, and logging. All the files that apply to a server are contained in a single server directory. The main configuration file for the server is called `config.xml`—this is where you configure the server's services and specify to which domain the server belongs.

An Oracle CEP *domain* is the management unit of a set of servers. There are two flavors of domains:

- Standalone-server domain—A domain that contains a single server. This is the type of domain created by default by the Configuration Wizard and is the starting point for a multi-server domain.

- Multi-server domain—A domain that contains two or more servers that share the same multicast address and port and share the security provider. Multi-server domains enable high availability for Oracle CEP applications. When you deploy an application to a multi-server domain, the application is replicated to each server in the domain.

The servers in a multi-server domain can be located on the same computer or on separate computers; what ties the servers together in a multi-server domain is that they have the same multicast address and port and belong to the same domain, all of which are configured in the server's `config.xml` file.

The following list describes the important files and directories of a server in a domain, relative to the server directory (which is a subdirectory of the main domain directory):

- `deployments.xml`—XML file that contains the list of applications, packaged as OSGi bundles, that are currently deployed to the Oracle CEP instance of this domain. You never update this file manually to deploy applications, but rather, use the deployer tool.
- `startwlevs.cmd`—Command file used to start an instance of Oracle CEP. The UNIX equivalent is called `startwlevs.sh`.
- `stopwlevs.cmd`—Command file used to stop an instance of Oracle CEP. The UNIX equivalent is called `stopwlevs.sh`.
- `config/config.xml`—XML file that describes the services that have been configured for the Oracle CEP instance. Services include logging, debugging, Jetty Web Service, and JDBC data sources.
- `config/security*`—Files that configure security for the domain.
- `config/atnstore.txt`—File that lists the configured users and groups for this domain.

You can also use the Configuration Wizard to update an existing server to reconfigure its administration user, listen ports, and JDBC configuration.

Creating a Standalone-Server Domain Using the Configuration Wizard

After you install Oracle CEP, use the Configuration Wizard to create a new domain to deploy your applications. The Configuration Wizard creates, by default, the domains in the *`ORACLE_CEP_HOME`*`/user_projects/domains` directory, where *`ORACLE_CEP_HOME`* refers to the Oracle CEP installation directory such as `d:/oracle_cep`. You can, however, create a domain in any directory you want.

The Configuration Wizard creates a single default server in the domain; all the server-related file are located in a subdirectory of the domain directory named the same as the server. Additionally, the Configuration Wizard allows you to:

- Configure the server's administration user and password.
- Configure the default server to use a database or database driver that is different from the default. In this case, you need to customize the JDBC settings to point to the appropriate database.
- Configure the server's listen port, the JMX RMI registry listen port, and the JMX RMI JRMP listen port.
- Configure the password for the identity keystore and private keystore.

You can use the Configuration Wizard in the following modes:

- **Graphical mode**

  Graphical-mode configuration is an interactive, GUI-based method for creating and configuring a domain. It can be run on both Windows and UNIX systems. See "Creating a Domain in Graphical Mode" on page 2-3.

- **Silent mode**

  Silent-mode configuration is a non-interactive method of creating and configuring a domain that requires the use of an XML properties file for selecting configuration options. You can run silent-mode configuration in either of two ways: as part of a script or from the command line. Silent-mode configuration is a way of setting configurations options only once and then using those options to duplicate the creating and configuration of a domain on many machines. See "Creating a Domain in Silent Mode" on page 2-5.

Creating a Domain in Graphical Mode

The following procedure shows how to invoke and use the Configuration Wizard in graphical mode by executing the relevant command script for both Windows or Unix. You can also invoke the Configuration Wizard on Windows using the Start menu:

```
Start > All Programs > Oracle Complex Event Processing 10gR3 > Tools >
Configuration Wizard
```

To invoke and use the Configuration Wizard in graphical mode, follow these steps:

1. Open a command window and set your environment as described in Setting Up Your Development Environment.

2. Change to the *ORACLE_CEP_HOME*/ocep_10.3/common/bin directory, where *ORACLE_CEP_HOME* refers to the main Oracle CEP installation directory, such as /oracle_cep:

   ```
   prompt> cd /oracle_cep/ocep_10.3/common/bin
   ```

3. Invoke the config.cmd (Windows) or config.sh (UNIX) command to invoke the wizard:

   ```
   prompt> config.sh
   ```

   After the Configuration Wizard has finished loading, you will see a standard Oracle Welcome window. Click Next.

   Note: The Oracle CEP Configuration Wizard is self-explanatory; however, if you want more information about using the tool, continue reading this procedure.

4. In the Choose Create or Update Domain window, choose Create a New Oracle CEP Domain. Click Next.

5. Enter the name of the administrator user for the default server of the domain. Click Next.

6. Enter basic configuration information about the default server in the domain. In particular:

   – Enter the name of the default server. This name will also be used as the name of the directory that contains the default server files.

   – The listen port for Oracle CEP itself. Default is 9002.

   – The secure listen port. Default is 9003.

   – The listen port for the JMX RMI registry, or the port on which to start the RMI registry. Default is 9004.

   – The listen port for JMX RMI JRMP, or the port on which to listen for RMI Java Remote Method Protocol (JRMP) JMX requests. Default is 9999.

   Click Next.

7. Enter and confirm the password for the Oracle CEP domain identity keystore. By default, the password for the certificate private key will be the same as the identity keystore; if you want it to be different, uncheck Use Keystore Password and enter the private key password. Click Next.

8. In the Configuration Options window, choose Yes if you want to change the default JDBC data source configuration, No to accept the defaults.

   The Configuration Wizard bases the creation of a new domain on the Oracle CEP domain template; by default, this template does not configure any JDBC data source for a domain. This means that, unless you change the default domain template used by the Configuration

Wizard, if you choose No at this step, *no* JDBC data source is configured. If you want to configure a JDBC data source, choose Yes at this step to proceed to the page in which you can enter the data source information.

Click Next.

9. If you chose to change the default JDBC data source configuration, enter the information in the Configure Database Properties window.

   In the top section, enter the name of the datasource. Then select the database type (Oracle or Microsoft SQL Server) and corresponding drivers; you can also browse to new drivers using the Browse/Append button.

   In the lower section, enter the details about the database to which this data source connects, such as its name, the name of the computer that hosts the database server, the port, and the name and password of the user that connects to the database. The JDBC connection URL is automatically generated for you based on this information.

   Click Next.

10. In the Configure Server window, enter the name of the new domain and the full pathname of its domain location. The configuration wizard creates the domain using its domain name in the domain location directory. Click Create.

    Note: Oracle recommends you always use the default domain location to create your domains: *`ORACLE_CEP_HOME`*`/user_projects/domains` (UNIX) or *`ORACLE_CEP_HOME`*`\user_projects\domains` (Windows).

11. If the creation of the domain succeeded, you will see a message similar to the following in the Creating Domain window:

    ```
    Domain created successfully!
    Domain location: C:\oracle_cep\user_projects\domains\wlevs30_domain
    ```

    Click Done.

Creating a Domain in Silent Mode

Using the Configuration Wizard in silent mode allows a non-interactive method of creating and configuring a domain; this method requires the use of an XML properties file for selecting configuration options. To run the Configuration Wizard using silent mode:

1. Create a `silent.xml` file that defines the configuration settings normally entered by a user during an interactive session of the Configuration Wizard. See "Creating a silent.xml File" on page 2-6.

Note: Incorrect entries in the `silent.xml` file can cause failures. To help you determine the cause of a failure, we recommend that you create a log file when you launch the Configuration Wizard.

2. Open a command window and change to the *`ORACLE_CEP_HOME`*`/ocep_10.3/common/bin` directory, where *`ORACLE_CEP_HOME`* refers to the main Oracle CEP installation directory, such as `/oracle_cep`:

   ```
   prompt> cd /oracle_cep/ocep_10.3/common/bin:
   ```

3. Invoke the `config.cmd` (Windows) or `config.sh` (UNIX) command in silent mode:

   ```
   prompt> config.cmd -mode=silent -silent_xml=path_to_xml_file
   ```

   where *`path_to_xml_file`* is the full pathname of the `silent.xml` template file you created in the preceding step.

   If you want to create an execution log, use the `-log=`*`full_path_to_log_file`* option; for example:

   ```
   prompt> config.cmd -mode=silent -silent_xml=path_to_xml_file
   -log=C:\logs\create_domain.log
   ```

   The command does not return any messages if it completes successfully. See "Returning Exit Codes to the Command Window" on page 2-9 for getting information about the success or failure of the silent execution of the Configuration Wizard.

Creating a silent.xml File

When you run the Configuration Wizard in silent mode, the program uses an XML file (`silent.xml`) to determine which configuration options should be used.

To create a `silent.xml` file, follow these steps:

1. Using your favorite XML editor, create an empty file called `silent.xml` on the computer on which you want to run the Configuration Wizard in silent mode.
2. Copy the contents of the sample XML file, shown in "Sample silent.xml File" on page 2-8, into your own `silent.xml` file.
3. In the `silent.xml` file you just created, edit the values for the keywords shown in Table 2-1 to reflect your configuration.

   For example, if you want to create the new domain in the `C:\oracle_cep\user_projects\domains` directory, update the corresponding `<data-value>` element as follows

```
<data-value name="DOMAIN_LOCATION"
            value="C:\oracle_cep\user_projects\domains" />
```

4. Save the file in the directory of your choice.

Table 2-1 Values for the silent.xml File

For this data-value name...	Enter the following value...
CONFIGURATION_OPTION	Specifies 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	Specifies the full pathname of an existing server in the domain. Use this option only when updating an existing server in a domain.
USERNAME	The username of the administrator of the created or updated server in the domain.
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 CEP domains is *`ORACLE_CEP_HOME`*`/user_projects/domains`, where *`ORACLE_CEP_HOME`* refers to the top-level installation directory, such as `c:/oracle_cep`.
NETIO_PORT	The port number to which the Oracle CEP server instance itself listens.
RMI_REGISTRY_PORT	The port on which to start the JMX RMI registry.

Table 2-1 Values for the silent.xml File

For this data-value name...	Enter the following value...
RMI_JRMP_PORT	The port on which to listen for RMI Java Remote Method Protocol (JRMP) JMX requests.
KEYSTORE_PASSWORD	The password for the Oracle CEP 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 datasource 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 datasource 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 datasource is configured for the server.

Sample 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_cep\user_projects\domains" />

    <data-value name="NETIO_PORT" value="9002" />
    <data-value name="RMI_REGISTRY_PORT" value="9004" />
    <data-value name="RMI_JRMP_PORT" value="9999" />
    <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>
```

Returning Exit Codes to the Command Window

When 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

Listing 2-1 provides a sample Windows command file that invokes the Configuration Wizard in silent mode and echoes the exit codes to the command window from which the script is executed.

Listing 2-1 Sample Windows Command File Displaying Silent-Mode Exit Codes

```
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.
```

Updating an Existing Server Using the Configuration Wizard

Use the Configuration Wizard to update an existing server in a domain. The procedure has similarities with creating a new domain and default server, so be sure you read "Creating a Standalone-Server Domain Using the Configuration Wizard" on page 2-2 before continuing with this section.

You can update the only following configuration options of an existing server in your domain:

- The listen port, the JMX RMI registry listen port, and the JMX RMI JRMP listen port.
- The configuration of the JDBC datasource.

For clarity, it is assumed in this section that you want to update a server called `productionServer` whose server-related files are located in the `C:\oracle_cep\user_projects\domains\mydomain\productionServer` directory.

Updating an Existing Server in Graphical Mode

Follow these steps to update an existing server in your domain using the Configuration Wizard in graphical mode.

1. Invoke the Configuration Wizard as described in "Creating a Domain in Graphical Mode" on page 2-3.

2. In the Choose Create or Update Domain window, choose “Update an existing Oracle CEP domain”. Click Next.
3. In the text box, enter the full pathname of the server directory that contains the files for the server you want to update. Following our example, this value would be `C:\oracle_cep\user_projects\domains\mydomain\productionServer`. Click Next.
4. Update the listen ports for the server. *Be sure that you do not enter the same values used by other servers in the domain so as to prevent any conflicts when all servers are running at the same time.* Click Next.
5. If you want to change the JDBC datasource configuration, select Yes, click Next, and enter the new values. Otherwise, select No and click Next.
6. Click Update to update the server.

Updating an Existing Server in Silent Mode

Updating an existing server in a domain in silent mode is similar to creating a new domain, as described in “Creating a Domain in Silent Mode” on page 2-5. The main difference is in the values of the options in the `silent.xml` file. In particular:

- Set CONFIGURATION_OPTION to `updateDomain`.
- Set EXISTING_DOMAIN_PATH to the full pathname of the server directory that contains the files for the server you want to update. In our example, the value would be `C:\oracle_cep\user_projects\domains\mydomain\productionServer`.
- 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.
- Se the listen port and JDBC datasource options to the new values.

  Be sure that the new server configuration options, such as NETIO_PORT, RMI_REGISTRY_PORT, and RMI_JRMP_PORT, are different than the options for any other servers in the domain. The database options can be the same if you want the updated server to connect to the same database as the other servers.

Based on the assumptions described in “Updating an Existing Server Using the Configuration Wizard” on page 2-10, the `silent.xml` file would look something like the following:

```
<?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_cep\user_projects\domains\mydomain\productionServer" />

  <data-value name="NETIO_PORT" value="9102" />
  <data-value name="RMI_REGISTRY_PORT" value="9104" />
  <data-value name="RMI_JRMP_PORT" value="9998" />

  <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>
```

Stopping and Starting the Server

Each Oracle CEP server directory contains a command script that starts a server instance; by default, the script is called `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX). The script to stop the server is called `stopwlevs.cmd` (Windows) or `stopwlevs.sh` (UNIX).

Starting the Server

To start an instance of Oracle CEP:

1. Ensure that the `JAVA_HOME` variable in the server start script points to the correct Oracle JRockit 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 of the HelloWorld domain is located in *`ORACLE_CEP_HOME`*`/ocep_10.3/samples/domains/helloworld_domain/defaultserver`, where *`ORACLE_CEP_HOME`* refers to the main Oracle CEP installation directory, such as `/oracle_cep`.

   If using the Oracle JRockit JDK installed with Oracle CEP 10.3, the `JAVA_HOME` variable should be set as follows:

   ```
   JAVA_HOME=ORACLE_CEP_HOME/jrockit-R27.6.0-50-1.6.0_05 (UNIX)

   set JAVA_HOME=ORACLE_CEP_HOME\jrockit-R27.6.0-50-1.6.0_05 (Windows)
   ```

   where *`ORACLE_CEP_HOME`* refers to the installation directory of Oracle CEP 10.3, such as `/oracle_cep` (UNIX) or `c:\oracle_cep` (Windows).

If using the Oracle JRockit JDK installed with Oracle JRockit Real Time 3.0, the `JAVA_HOME` variable should be set as follows:

```
JAVA_HOME=ORACLE_RT_HOME/jrrt-3.0.0-1.6.0 (UNIX)
set JAVA_HOME=ORACLE_RT_HOME\jrrt-3.0.0-1.6.0 (Windows)
```

where *`ORACLE_RT_HOME`* refers to the installation directory of Oracle JRockit Real Time 3.0, such as `/jrockit` (UNIX) or `c:\jrockit` (Windows).

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

   ```
   prompt> cd
   C:\oracle_cep\ocep_10.3\samples\domains\helloworld_domain\defaultserver
   ```

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

   ```
   prompt> startwlevs.cmd
   ```

   If you are using the Oracle JRockit JDK included in Oracle JRockit Real Time 3.0, enable the deterministic garbage collector by passing the `-dgc` parameter to the command:

   ```
   prompt> startwlevs.cmd -dgc
   ```

Stopping the Server Using the stopwlevs Script

To stop a running Oracle CEP server instance:

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

   ```
   prompt> cd
   C:\oracle_cep\ocep_10.3\samples\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:rmi:///jndi/rmi://`*`host:jmxport`*`/jmxrmi`, where *`host`* refers to the computer hosting the server and *`jmxport`* refers to the server's JMX port, configured in `config.xml` file. For example:

   ```
   prompt> stopwlevs.sh -url
   service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
   ```

   In the example, the host is `ariel` and the JMX port is `9004`.

   See Table 5-1, "Connection Arguments," on page 5-9 for additional details about the `-url` argument.

Next Steps

After creating your own Oracle CEP standalone-server domain, you can:

- Optionally configure the server. See "Overview of Configuring Oracle CEP Servers" on page 3-1.
- Create additional servers in the domain and configure the domain to be multi-server. See "Configuring and Using Oracle CEP Multi-Server Domains" on page 4-1.
- Create an Oracle CEP application. See Oracle CEP Application Development Guide for a description of the programming model, details about the various components that make up an application, how they all fit together, and typical steps to create a new application.
- Deploy your new, or existing, Oracle CEP application to the domain. See Deploying Oracle CEP Applications.

CHAPTER 3

Configuring Oracle CEP Servers

This section contains information on the following subjects:

Overview of Configuring Oracle CEP Servers

After you have created an Oracle Complex Event Processing (or *Oracle CEP* for short) domain along with at least a single server, you start a server instance so you can then deploy applications and begin running them. See "Stopping and Starting the Server" on page 2-12 for details.

There are a variety of ways to configure a particular server instance, as follows:

- Update the server configuration file, `config.xml`, manually. You

  See "Configuring the Server by Manually Editing the config.xml File" on page 3-2.

- Use the `wlevs.Admin` utility to administer a Oracle CEP instance and to dynamically configure the EPL rules for the processors of a deployed application.

  See "wlevs.Admin Command-Line Reference" on page 5-1.

- Use the Visualizer Administration Console, which is a Web 2.0 application that consumes data from Oracle CEP, displays it in a useful and intuitive way to system administrators and operators, and, for specified tasks, accepts data that is then passed back to Oracle CEP so as to change it configuration.

  See Overview of Visualizer.

- Use standards-based interfaces that are fully compliant with the Java Management Extensions (JMX) specification to change the configuration of the domain and deployed applications.

  See "Managing Applications, Servers, and Domains Using MBeans" on page 6-1 and the Javadoc for details about the Oracle CEP MBeans.

Configuring the Server by Manually Editing the config.xml File

The Oracle CEP server configuration file, `config.xml`, is located in the *`DOMAIN_DIR`*`/`*`servername`*`/config` directory, where *`DOMAIN_DIR`* refers to the main domain directory and *`servername`* refers to the name of the particular server instance in the domain. To change the configuration of an Oracle CEP server instance, you update this file

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 multi-server domain are configured together. This includes the multicast address and port, the groups, and so on.

  See "Configuring and Using Oracle CEP Multi-Server Domains" on page 4-1.

- Jetty, an open-source, standards-based, full-featured Java Web Server.

  See "Configuring Jetty for Oracle Complex Event Processing" on page 8-1.

- JDBC data source, used to connect to a relational database.

  See "Configuring Access to a Relational Database" on page 10-1.

- JMX, required to use the `wlevs.Admin` utility.

  See "Configuring JMX for Oracle Complex Event Processing" on page 9-1.

- HTTP publish-subscribe server.

  See "Configuring the HTTP Publish-Subscribe Server" on page 11-1.

- Logging and debugging properties of the server. By default, the log security level is set to `NOTICE`.

  See "Configuring Logging and Debugging" on page 12-1.

The following sample `config.xml`, from the *ORACLE_CEP_HOME*`/user_projects/domains/wlevs30_domain/defaultserver` template domain, shows how to configure 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>
      <rmi-jrmp-port>9999</rmi-jrmp-port>
      <jndi-service-name>JNDI</jndi-service-name>
      <rmi-registry-port>9004</rmi-registry-port>
   </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>wlevs30_domain</name>
</domain>
```

WARNING: If you update the `config.xml` file manually to change the configuration of Oracle CEP, you must restart the server for the change to take effect.

CHAPTER 4

Configuring and Using Oracle CEP Multi-Server Domains

This section contains information on the following subjects:

Overview of Oracle CEP Multi-Server Domains

The following sections introduce use cases and terminology related to the management and availability of a set of Oracle CEP servers.

Oracle CEP Multi-Server Domain

An Oracle CEP multi-server domain is a set of two or more servers logically connected for the purposes of management, and physically connected using a shared User Datagram Protocol (UDP) multicast address and port. All servers in an Oracle CEP multi-server domain are aware

of all other servers in the domain and any one server can be used as an access point for making changes to the deployments in the domain.

Management of the multi-server infrastructure is done at the domain level. Thus server failure, start, or restart is detected by every member of the multi-server domain. Each member of the multi-server domain has a consistent, agreed notion of domain membership enforced by the multi-server infrastructure.

For servers to be considered a part of the same multi-server domain they must share the same multicast address and port and the same domain name.

The servers of a multi-server domain must be homogenous. For example, you cannot have two servers, one configured with the IPv6 (Internet Protocol version 6) protocol and the other with the IPv4 protocol, be members of the same domain.

Groups

In order to support the deployment to, and management of, the multi-server domain at a finer grained-level than the domain, Oracle CEP introduces the concept of *groups*. A group is a set of one or more servers with a unique name within the domain. In an Oracle CEP domain, an arbitrary number of groups may exist with a configurable group membership. A server may be a member of more than one group, although typically this information is transparent to the user. The following pre-defined groups always exist:

- **Singleton server group**—Consists of only the local server. This means that the membership of this group depends on the server from which it is accessed. This group can be used to pin deployments to a single server.
- **Domain (or default) group**—Contains all live members of the domain. Its membership is automatically managed and cannot be changed by the user.

When you deploy an application to a multi-server domain, you deploy it to a particular group. Applications deployed to the domain or custom groups must have a unique name across the domain.

Multi-Server Notifications and Messaging

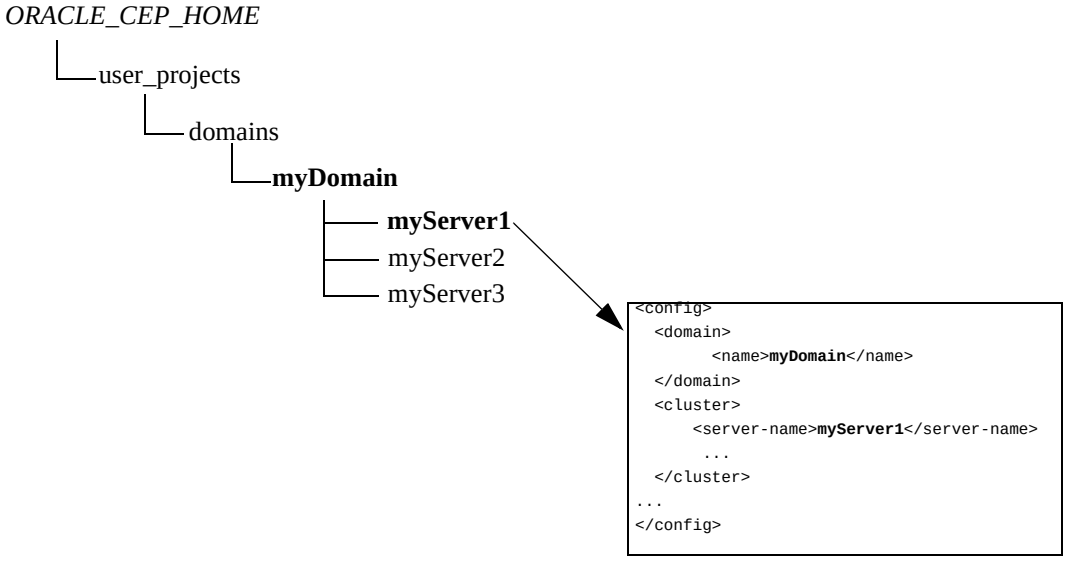
In order to provide high availability (HA)-like capabilities to adapter and event beam implementations, Oracle CEP provides a number of notification and messaging APIs at both the group- and domain-level. Using these APIs, you can configure a server to receive notification when its group or domain membership changes, either because an administrator deliberately

changed it or due to a server failure. Similarly you can use these APIs to send messages to both individual groups and to the domain.

Multi-Server Domain Directory Structure

Servers in an Oracle CEP domain store their files in a single directory. By convention, the directories of the servers in a multi-server domain are sub-directories of the domain directory. Additionally, 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 multi-server domain are located on different computers, you can replicate the directory structure on both computers, also for simplicity and consistency.

The following diagram shows a multi-server domain directory with three servers. A snippet of the configuration file for `myServer1` is shown to show how the domain directory and domain object are configured with the same name, as well as the server directory and server name. The domain directory is located in the *`ORACLE_CEP_HOME`*`/user_projects/domains` directory, which is the default location for Oracle CEP domains.



Creating and Configuring a Simple Multi-Server Domain

This section describes how to create and configure a simple multi-server domain from two or more Oracle CEP servers. The multi-server domain is simple because it is not configured with any custom groups, other than the two predefined ones (singleton group and domain group.) See "Configuring Custom Groups for a Multi-Server Domain" on page 4-9 for a description of how to configure custom groups within the multi-server domain.

Note: In this section it is assumed that you have already created a domain that contains a single server and that you want to add additional servers to the domain to make it a multi-server domain. See "Creating and Updating an Oracle CEP Standalone-Server Domain" on page 2-1 for details on creating a domain.

The following high level steps describe configuring a multi-server domain:

1. Add one or more servers to the domain using the Configuration Wizard.

   Note: Even though the Configuration WIzard does not support adding new servers to a multi-server domain, one can use the Configuration Wizard to generate a new stand-alone server, and then manually update its configuration to join a multi-server domain.

   See "Adding New Servers to an Existing Domain Using the Configuration Wizard" on page 4-4.

2. Configure all the servers in the multi-server domain by manually editing their `config.xml` files and adding a `<cluster>` element with specific information.

   See "Configuring the Servers in a Multi-Server Domain" on page 4-7 for details.

Adding New Servers to an Existing Domain Using the Configuration Wizard

Use the Configuration Wizard to add a new server to an existing standalone server domain so as to later convert it into a multi-server domain. The procedure is similar to creating a new domain, so be sure you read "Creating a Standalone-Server Domain Using the Configuration Wizard" on page 2-2 before continuing with this section.

For clarity, it is assumed that:

- You have already created a new domain and its domain directory is `C:\oracle_cep\user_projects\domains\mydomain`.

- The domain includes a single server called `defaultserver` and the server files are located in the `C:\oracle_cep\user_projects\domains\myDomain\myServer1` directory.
- You want to create a new server in the existing `mydomain` domain called `myServer2`.

You can use the Configuration Wizard in graphical or silent mode.

Adding New Servers in Graphical Mode

Follow these steps to add a new server to an existing domain in graphical mode:

1. Invoke the Configuration Wizard as described in "Creating a Domain in Graphical Mode" on page 2-3.
2. In the Choose Create or Update Domain window, choose Create a New Oracle CEP Domain. Click Next.
3. Enter the name and password of the administrator user for the new server you are adding to the domain. Click Next.
4. Enter basic configuration information about the new server in the domain. ***If the new server is located on the same computer as any other servers in the domain, be sure the following information is different from that of the other servers to prevent conflicts when starting all servers***. In particular:
   - Enter the name of the new server. This name will also be used as the name of the directory that contains the new server's files. Following our example, this value is `myServer1`.
   - The listen port for Oracle CEP itself.
   - The listen port for the JMX RMI registry, or the port on which to start the RMI registry.
   - The listen port for JMX RMI JRMP, or the port on which to listen for RMI Java Remote Method Protocol (JRMP) JMX requests.

   Click Next.
5. Enter and confirm the password for the Oracle CEP identity keystore. By default, the password for the certificate private key will be the same as the identity keystore; if you want it to be different, uncheck Use Keystore Password and enter the private key password. Click Next.
6. In the Configuration Options window, choose Yes if you want to change the default JDBC data source configuration, No to accept the defaults. Click Next.

NOTE: When you deploy an application to a group in the domain, Oracle CEP replicates the application to each server that is a member of the group. This means that if your application uses a datasource, and you have configured the datasource differently for each server in the domain, then the storage and retrieval of data to and from this data source will differ depending on the server on which the application is running.

7. If you chose to change the default JDBC data source configuration, enter the information in the Configure Database Properties window.

   In the top section, enter the name of the datasource. Then select the database type (Oracle or Microsoft SQL Server) and corresponding drivers; you can also browse to new drivers using the Browse/Append button.

   In the lower section, enter the details about the database to which this data source connects, such as its name, the name of the computer that hosts the database server, the port, and the name and password of the user that connects to the database. The JDBC connection URL is automatically generated for you based on this information.

   Click Next.

8. In the Configure Server window, enter the name of the *existing* domain and the full pathname of its location. Following our example, you would enter `myDomain` for the domain name and `C:\oracle_cep\user_projects\domains` for the domain location. Click Create.

9. If the creation of the new server succeeded, you will see a message similar to the following in the Creating Domain window:

   ```
   Domain created successfully!
   Domain location: C:\oracle_cep\user_projects\domains\myDomain
   ```

   Click Done.

Adding New Servers in Silent Mode

Adding a new server to an existing domain in silent mode is similar to creating a new domain, as described in "Creating a Domain in Silent Mode" on page 2-5. The only difference is in the values of the options in the `silent.xml` file. In particular:

- Set CONFIGURATION_OPTION to `createDomain`.
- 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_cep\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 `myServer1`.

- If this server is running on the same computer as the other servers in the multi-domain, then be sure that the new server configuration options, such as NETIO_PORT, RMI_REGISTRY_PORT, and RMI_JRMP_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 multi-server domain, then the ports do not have to be different.

Based on the assumptions described in "Adding New Servers to an Existing Domain Using the Configuration Wizard" on page 4-4, the silent.xml file would look something like the following:

```
<?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_cep\user_projects\domains" />

    <data-value name="NETIO_PORT" value="9102" />
    <data-value name="RMI_REGISTRY_PORT" value="9104" />
    <data-value name="RMI_JRMP_PORT" value="9998" />
    <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>
```

Configuring the Servers in a Multi-Server Domain

To configure the servers in a multi-server domain, update the `config.xml` file for *each member server* by adding a `<cluster>` child element of the root `<config>` element. Include the `<server-name>`, `<multicast-address>`, `<identity>`, and `<enabled>` child elements of `<cluster>`. The order of the elements is important; see "Order and Additional Child Elements

of the <cluster> Element" on page 4-15 for details. The following example shows a possible configuration:

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

In the example, the server is part of a domain called `myDomain`.

For each server of the multi-server 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 multi-server domain.

The `<multicast-address>` element is required unless all servers of the multi-server domain are hosted on the same computer; in that case you can omit the `<multicast-address>` element and Oracle CEP automatically assigns a multicast address to the multi-server 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.

The `<identity>` element identifies the server's identity and must be an integer between 1 and INT_MAX. Oracle CEP numerically compares the server identities during multi-server operations; the server with the lowest identity becomes the domain coordinator. Be sure that each server in the multi-server domain has a different identity; if servers have the same identity, the results of multi-server operations are unpredictable.

The `<server-name>` child element of `<cluster>` specifies a unique name for the server. 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>`.

Finally, by default the clustering of the servers in a multi-server domain is disabled, so you must explicitly enable it with the `<enabled>` element.

An example of configuring a second server, called `myServer2`, in the `myDomain` multi-server domain is shown below; note that its identity 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>true</enabled>
  </cluster>

...

</config>
```

See "Order and Additional Child Elements of the <cluster> Element" on page 4-15 for a brief description of additional multi-server-related configuration elements and the required order of child elements.

Configuring Custom Groups for a Multi-Server Domain

There are cases where the application logic cannot simply be replicated across a homogenous set of servers in a multi-server domain. Examples of these types of applications are those that must determine the best price provided by different pricing engines, or applications that send an alert when a position crosses a threshold. In these cases, the application is not idempotent; it must calculate only once or send a single event. In other cases, the application has a singleton nature, such as a monitoring application, the HTTP pub-sub server, and so on.

As a more complex example, consider a domain that has two applications: the `strategies` application uses several strategies for calculating different prices for some derivative and then feeds its results to a `selector` application. The `selector` application then selects the best price amongst the different options provided by the `strategies'` application results. The `strategies` application can be replicated to achieve fault-tolerance. However, the `selector` application must be able to keep state so as to determine the best price; for this reason, the `selector` application *cannot* be replicated in a hot/hot fashion.

For all these reasons, a domain must support servers that are not completely homogeneous; you configure this by creating custom groups.

To configure a group, update the `config.xml` file of each member of the group, adding (if one does not already exist) a `<groups>` child element of `<cluster>` and specifying the name of the group as the value of the `<groups>` element. The `<groups>` element can include more than one group name in the case that the server is a member of more than one group; separate multiple group names using commas. The `<groups>` element is optional; if a server configuration does not include one, then the server is a member of the default groups (domain and singleton).

For example, assume you have created three servers (`myServer1`, `myServer2`, and `myServer3`) and you want `myServer1` to be a member of the `selector` group and `myServer2` and `myServer3` to be members of the `strategy` group. The relevant snippets of the `config.xml` file for each server are shown below:

Listing 4-1 Config.xml of 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>true</enabled>
      <groups>selector</groups>
  </cluster>
...
</config>
```

Listing 4-2 Config.xml of 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>true</enabled>
      <groups>strategy</groups>
  </cluster>
...
</config>
```

Listing 4-3 Config.xml of 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>true</enabled>
      <groups>strategy</groups>
  </cluster>
...
</config>
```

See "Order and Additional Child Elements of the <cluster> Element" on page 4-15 for a brief description of additional multi-server-related configuration elements and the required order of child elements.

Starting the Servers in a Multi-Server Domain

To start the servers in a multi-server 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 "Stopping and Starting the Server" on page 2-12 for details.

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

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

Deploying Applications to a Multi-Server Domain

When you deploy an application to a multi-server domain, you typically specify a target group, and Oracle CEP then deploys the application to the set of running servers in that group. Oracle CEP dynamically maintains group membership based on running servers. This means that if new servers in the group are started, Oracle CEP automatically propagates the appropriate set of deployments to the new server.

Take, for example, the simple multi-server domain configured in the section "Creating and Configuring a Simple Multi-Server Domain" on page 4-4. Assume that only `myServer1` had been started, and then an application is deployed to the domain group, which includes `myServer1` and `myServer2`. At that point, because only `myServer1` of the multi-server domain has been started, the application will be deployed only to `myServer1`. When `myServer2` is subsequently started, Oracle CEP *automatically* replicates and propagates the deployment of the application to `myServer2` without the user having the explicitly deploy it.

Deployment propagation only occurs when a server is missing a required deployment. If a server already has a local deployment, then this deployment is used. This means that Oracle CEP does not automatically propagate *changes* to a deployment on one server of the multi-server domain to the other servers if they already have that deployment. This means that it is possible to have slightly different versions of the same deployment on different servers if the deployment is configured manually through copying application jar files.

If different configuration is required on different servers for an application then currently it is best to achieve this by using system properties.

For complete reference on the Deployer command-line tool, see Deployer Command-Line Reference.

Deploying to the Singleton Server Group

If you do not specify a group when you deploy an application, Oracle CEP 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 single-server domains, but is also applicable to multi-server domains.

Note: When you upgrade a 2.0 domain to execute in a multi-server domain, any deployed applications are deployed to the singleton server group.

The following example shows how to deploy to a singleton group; note that the command does not specify a `-group` option:

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

In the example, the `myapp_1.0.jar` application will be deployed to the singleton server group that contains a single server: the one running on host `ariel` and listening to port `9002`. If the domain is multi-server and other servers are members of the domain group, the application will *not* be deployed to these servers.

Deploying to the Domain Group

The domain group is a live group that always exists and contains all servers in a domain. In another words, 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 consistency in usage.

When you explicitly deploy an application to the domain group, Oracle CEP 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:

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

In the example, the `myapp_1.0.jar` application will be deployed to all servers of the domain group on host `ariel` listening to port `9002`.

Deploying to a Custom Group

To deploy to a custom group, use the `-group` *`groupname`* option of the deploy command.

In the following examples, assume the multi-server domain has been configured as described in "Configuring Custom Groups for a Multi-Server Domain" on page 4-9.

The following example shows how to deploy an application called `strategies_1.0.jar` to the `strategygroup`:

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

Based on the multi-server domain configuration, the preceding command deploys the application to `myServer2` and `myServer3`, the members of the group `strategygroup`.

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

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

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

Note that both commands are executed to the same server (the one on host `ariel` listening to port `9002`). However, 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.

Using the Multi-Server Domain APIs to Manage Group Membership Changes

In an active-active system, applications are deployed homogeneously across several servers and are actively executing.

There are cases, however, when these homogeneously-deployed applications need to elect a primary one as the coordinator or leader. In this case, events that result from the coordinator application are kept and passed on to the next component in the EPN; the results of secondary servers are dropped. However, if the coordinator fails, then one of the secondary servers must be elected as the new coordinator.

To enable this in an application, the adapter or event bean, generally in the role of an event sink, must implement the `com.bea.wlevs.ede.api.cluster.GroupMembershipListener` interface which allows the event sinks to listen for multi-server domain group membership

changes. At runtime, Oracle CEP automatically invokes the `onMembershipChange` callback method whenever membership changes occur.

The signature of the callback method is as 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()`.

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

A similar interface `com.bea.wlevs.ede.api.cluster.DomainMembershipListener` exists for listening to membership changes to the domain as a whole, rather than just changes to the group.

Order and Additional Child Elements of the <cluster> Element

As specified by the Server Configuration XSD Schema, the order of the child elements of the <cluster> element in the config.xml file is important; if you include elements in the incorrect order you may encounter an error. The following lists describes the order in which you should list the child elements; see the end of this section for information about elements that have not yet been discussed:

- `<server-name>`
- `<server-host-name>`
- `<multicast-address>`
- `<multicast-port>`
- `<identity>`
- `<enabled>`
- `<security>`
- `<groups>`

- `<operation-timeout>`

The preceding sections discuss some of the child elements of the `<cluster>` element of the `config.xml` file, in particular `<server-name>`, `<multicast-address>`, `<identity>`, `<groups>`, and `<security>`.

This section briefly describes additional child elements. For the complete XSD Schema of the config.xml file, including a description of the `<cluster>` element, see Server Configuration XSD Schema.

You can add the following optional child elements to the `<cluster>` element of the `config.xml` file to further configure your multi-server domain:

- `<server-host-name>`—Specifies the host address/IP used for point-to-point HTTP multi-server communication. Default value is `localhost`.
- `<multicast-port>`—Specifies the port used for multicast traffic. Default value is `9005`.
- `<operation-timeout>`—Specifies, in milliseconds, the timeout for point-to-point HTTP multi-server requests. Default value is `30000`.

Troubleshooting Multi-Server Domains

Question: After I deploy my application to a multi-server domain, Oracle CEP stops it after about 30 seconds.

Answer: Be sure you do not have more than one VPN software package installed on the same computer hosting your multi-server domain.

CHAPTER 5

wlevs.Admin Command-Line Reference

The following sections describe the `wlevs.Admin` utility:

Overview of the wlevs.Admin Utility

The `wlevs.Admin` utility is a command-line interface to administer Oracle Complex Event Processing (or *Oracle CEP* for short) and, in particular, dynamically configure the EPL rules for application processors and monitor the event latency and throughput of an application. The utility internally uses JMX to query the configuration and runtime MBeans of both the server and deployed applications.

The Oracle CEP configuration framework allows concurrent changes to both the application and server configuration by multiple users. The framework does not use locking to manage this concurrency, but rather 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 is allowed to commit their changes. The other user will then get an error if they try to update the same configuration object, and must refresh their session to view the updated configuration.

Each command of the `wlevs.Admin` utility 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, you must use JMX directly to make these changes rather than the `wlevs.Admin` utility.

Required Environment for the wlevs.Admin Utility

To set up your environment for the `wlevs.Admin` utility:

1. Install and configure the Oracle CEP software, as described in the Oracle CEP *Installation Guide*.
2. Configure JMX connectivity for the domain you want to administer. See "Configuring JMX for Oracle Complex Event Processing" on page 9-1.
3. Open a command window and set your environment as described in Setting Up Your Development Environment.
4. 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 *`ORACLE_CEP_HOME`*`/ocep_10.3/bin/wlevsadmin.jar` file, where *`ORACLE_CEP_HOME`* refers to the main Oracle CEP installation directory. When you use the `-jar` argument, you do not specify the `wlevs.Admin` utility name at the command line. For example

     ```
     prompt> java -jar d:/oracle_cep/ocep_10.3/bin/wlevsadmin.jar
     -url service:jmx:rmi:///jndi/rmi://localhost:9004/jmxrmi
     ```

```
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_CEP_HOME*/ocep_10.3/bin/wlevsadmin.jar

  • *ORACLE_CEP_HOME*/ocep_10.3/bin/wlevs.jar

  • *ORACLE_CEP_HOME*/ocep_10.3/modules/com.bea.wlevs.deployment.server_3.0.0.0.jar

  • *ORACLE_CEP_HOME*/ocep_10.3/modules/com.bea.wlevs.ede_3.0.0.0.jar

  • *ORACLE_CEP_HOME*/ocep_10.3/modules/com.bea.wlevs.management_3.0.0.0.jar

  • *ORACLE_CEP_HOME*/modules/com.bea.core.jmx_6.0.0.0.jar

  • *ORACLE_CEP_HOME*/modules/com.bea.core.jndi.context_6.0.0.0.jar

  • *ORACLE_CEP_HOME*/modules/com.bea.core.rmi_6.0.0.0.jar

  • *ORACLE_CEP_HOME*/modules/com.bea.core.i18n_1.4.0.0.jar

  • *ORACLE_CEP_HOME*/modules/com.bea.core.diagnostics.core_2.1.0.0.jar

  • *ORACLE_CEP_HOME*/modules/javax.xml.stream_1.1.1.0.jar

  where *ORACLE_CEP_HOME* refers to the main directory into which you installed Oracle CEP.

Running the wlevs.Admin Utility Remotely

Sometimes it is useful to run the `wlevs.Admin` utility on a computer different from the computer on which Oracle CEP is installed and running. To run the utility remotely, follow these steps:

1. Copy the following JAR files from the computer on which Oracle CEP 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_CEP_HOME*/ocep_10.3/bin/wlevsadmin.jar

   – *ORACLE_CEP_HOME*/ocep_10.3/modules/com.bea.wlevs.deployment.server_3.0.0.0.jar

   – *ORACLE_CEP_HOME*/ocep_10.3/modules/com.bea.wlevs.ede_3.0.0.0.jar

- *ORACLE_CEP_HOME*/ocep_10.3/modules/com.bea.wlevs.management_3.0.0.0.jar
- *ORACLE_CEP_HOME*/modules/com.bea.core.jmx_6.0.0.0.jar
- *ORACLE_CEP_HOME*/modules/com.bea.core.jndi.context_6.0.0.0.jar
- *ORACLE_CEP_HOME*/modules/com.bea.core.rmi_6.0.0.0.jar
- *ORACLE_CEP_HOME*/modules/com.bea.core.i18n_1.4.0.0.jar
- *ORACLE_CEP_HOME*/modules/com.bea.core.diagnostics.core_2.1.0.0.jar
- *ORACLE_CEP_HOME*/modules/javax.xml.stream_1.1.1.0.jar

where *ORACLE_CEP_HOME* refers to the main directory into which you installed Oracle CEP.

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. Invoke the `wlevs.Admin` utility as described in the next section.

Running wlevs.Admin Utility in SSL Mode

To use SSL when using the `wlevs.Admin` command-line utility, you must first create a trust keystore, as described in the following steps:

1. Open a command window and set your environment as described in Setting Up Your Development Environment.
2. If not already running, start the Oracle CEP server.
3. Change to the *DOMAIN_DIR*/*servername*/`ssl` directory, where *DOMAIN_DIR* refers to the main domain directory and *servername* refers to the name of your server, such as `d:\oracle_cep\user_projects\domains\mydomain\myserver\ssl`.
4. Generate a trust keystore by specifying the following command:

```
prompt> java -jar
ORACLE_CEP_HOME/ocep_10.3/utils/security/wlevsgrabcert.jar
host:secureport truststorepath trustfile truststorepassword password
```

where

- *ORACLE_CEP_HOME* refers to the directory in which you installed Oracle CEP, such as `d:\oracle_cep`.
- *host* refers to the host on which Oracle CEP is running
- *secureport* refers to the network i/o port configured for SSL; see "How SSL Is Configured in Oracle CEP" on page 7-20.
- *trustfile* refers to the name of the generated trust keystore file; default is `evstrust.jks`
- *password* refers to the password you want to assign to the trust keystore file; default is `changeit`.

For example:

```
prompt> java -jar /oracle_cep/ocep_10.3/utils/security/wlevsgrabcert.jar
ariel:9003 truststorepath clitrust.jks truststorepassword secret
```

To specify that the `wlevs.Admin` command-line utility use this trust keystore file, use the following properties:

- `-Djavax.net.ssl.trustStore`—Name of the trust keystore file you created in the preceding step
- `-Djavax.net.ssl.trustStorePassword`—Password of the trust keystore file.

Also be sure to specify the secure port in the URL. For example:

```
prompt> java wlevs.Admin
        -Djavax.net.ssl.trustStore=clitrust.jks
        -Djavax.net.ssl.trustStorePassword=secret
        -url service:jmx:rmi:///jndi/rmi://ariel:9003/jmxrmi
        -username wlevs -password wlevs
        SHUTDOWN -scheduleAt 600
```

Syntax for Invoking 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" on page 5-7
- "User Credentials Arguments" on page 5-10
- "Common Arguments" on page 5-11

The following sections provide detailed syntax information about the supported commands of the `wlevs.Admin` utility:

- "Commands for Managing the Server Life Cycle" on page 5-13
- "Commands for Managing the EPL Rules of an Application" on page 5-14
- "Commands for Managing Oracle CEP MBeans" on page 5-29
- "Commands for Controlling Event Record and Playback" on page 5-38
- "Commands for Monitoring Throughput and Latency" on page 5-47

Example Environment

In many of the examples throughout the sections that follow, it is assumed that a certain environment has been set up:

- The Oracle CEP instance listens to JMX requests on port `9004`.
- The Oracle CEP instance uses the name of its host machine, `ariel`, as its listen address.
- The `wleves` username has system-administrator privileges and uses `wlevs` for a password.

Also, for clarity, all the examples are shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

Exit Codes Returned by wlevs.Admin

All `wlevs.Admin` commands return an exit code of `0` if the command succeeds and an exit code of `1` if 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)` if an exception is raised while processing a command, causing Ant and other Java client JVMs to exit.

Connection Arguments

```
java wlevs.Admin
     [ {-url URL} | {-listenAddress hostname -listenPort port} ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     COMMAND-NAME command-arguments
```

When you invoke most `wlevs.Admin` commands, you specify the arguments in Table 5-1 to connect to an Oracle CEP instance.

Table 5-1 Connection Arguments

Argument	Definition

`-url service:jmx:rmi:///jndi/rmi://`*`host:jmxport`*`/jmxrmi`	Specifies the URL that establishes a JMX connection to the Oracle CEP instance you want to administer, where:
	• *`host`* refers to the name of the computer on which the Oracle CEP instance is running.
	• *`jmxport`* refers to the port configured for Oracle CEP that listens to JMX connections.
	This port is configured in the `config.xml` file of the Oracle CEP domain you are administering. In particular, you specify the port using the `<rmi-registry-port>` child element of the `<jmx>` element, as shown:
	`<jmx>`
	`  <jndi-service-name>JNDI</jndi-service-name>`
	`  <rmi-service-name>RMI</rmi-service-name>`
	`<rmi-registry-port>9004</rmi-registry-port>`
	`  <rmi-jrmp-port>9999</rmi-jrmp-port>`
	`</jmx>`
	In the example, the JMX port is `9004`.
	If you use this argument, do not specify `-listenAddress` or `-listenPort`.
	Other than *`host`* and *`jmxport`*, you specify the remainder of the URL as written.
	For example, if Oracle CEP is running on a computer with hostname ariel, and the JMX listening port is 9004, then the URL would be:
	`-url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi`
	See "Configuring JMX for Oracle Complex Event Processing" on page 9-1 for details about configuring JMX, JNDI, and RMI for Oracle CEP.

Table 5-1 Connection Arguments (Continued)

Argument	Definition
`-listenAddress` *`hostname`*	Specifies the name of computer on which the Oracle CEP 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 CEP is running on a computer with hostname `ariel`, then this argument would be: `-listenAddress ariel`

`-listenPort` *`port`*	Specifies the port configured for Oracle CEP 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 `-listenPort`, *instead* of `-url`.
	The JMX port is configured in the `config.xml` file of the Oracle CEP domain you are administering. In particular, the port is the `<rmi-registry-port>` child element of the `<jmx>` element, as shown:
	`<jmx>`
	`  <jndi-service-name>JNDI</jndi-service-name>`
	`  <rmi-service-name>RMI</rmi-service-name>`
	`  **<rmi-registry-port>9004</rmi-registry-port>**`
	`  <rmi-jrmp-port>9999</rmi-jrmp-port>`
	`</jmx>`
	In the example, the JMX port is `9004` and you specify as an argument as follows:
	`-listenPort 9004`
	See "Configuring JMX for Oracle Complex Event Processing" on page 9-1 for details about configuring JMX, JNDI, and RMI for Oracle CEP.

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 specify the arguments in Table 5-2 to provide the user credentials of an Oracle CEP user who has permission to invoke the command.

If security has not been enabled for your Oracle CEP domain, then you do not have to provide user credentials.

Table 5-2 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 if the `wlevs.Admin` utility cannot connect to the server or if the Oracle CEP instance rejects the username and password.

Common Arguments

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ -verbose ]
     COMMAND-NAME command-arguments
```

All `wlevs.Admin` commands support the argument in Table 5-3 to get verbose output.

Table 5-3 Common Arguments

Argument	Definition
`-verbose`	Specifies that `wlevs.Admin` should output additional verbose information.

Command for Getting Usage Help

HELP

Provides syntax and usage information for all Oracle CEP commands (by default) 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 the Oracle CEP is installed. You do not need to start a server instance to invoke this command, nor do you need to supply user credentials, even if 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.
- An actual 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:

```
prompt> java wlevs.Admin HELP UPLOAD
```

The command returns the following:

```
Description:
Uploads rules to be configured in the EPL Processor.

Usage:
java wlevs.Admin
   [-url | -listenAddress <host-name> -listenPort <port>]
  -username <username> -password <password>
 UPLOAD -application <application name> -processor <eplprocessor name>
-sourceURL "source url"

Where:
-application = Name of the application.
-processor = Name of the EPL Processor.
-sourceURL = source URL containing the rules in an XML format.

java wlevs.Admin -url service:jmx:rmi:///jndi/rmi://localhost:9004/jmxrmi
-username wlevs -password wlevs UPLOAD -application myapplication -processor
eplprocessor -sourceURL file:///d:/test/rules.xml
```

Commands for Managing the Server Life Cycle

Table 5-4 is an overview of commands that manage the life cycle of a server instance. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table 5-4 Overview of Commands for Managing the Server Life Cycle

Command	Description
SHUTDOWN	Gracefully shuts down a WebLogic Event Server.

SHUTDOWN

Gracefully shuts down the specified Oracle CEP instance.

A graceful shutdown gives Oracle CEP time to complete certain application processing currently in progress.

The `-url` connection argument specifies the particular Oracle CEP instance that you want to shut down, based on the `host` and `jmxport` values. See "Connection Arguments" on page 5-7 for details.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     SHUTDOWN [-scheduleAt seconds]
```

Table 5-5 SHUTDOWN Arguments

Argument	Definition
`-scheduleAt` *seconds*	Specifies the number of seconds after which the Oracle CEP instance shuts down. If you do not specify this parameter, the server instance shuts down immediately.

Example

The following example instructs the specified Oracle CEP instance to shut down in ten minutes:

```
prompt> java wlevs.Admin
       -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
       -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.

Commands for Managing the EPL Rules of an Application

Table 5-6 is an overview of commands that manage the EPL rules for a particular processor of an Oracle CEP application. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table 5-6 Overview of Commands for Managing Application EPL Rules

Command	Description
ADDRULE	Adds a new EPL rule to the processor of an Oracle CEP application.
DELETERULE	Deletes an existing EPL rule from the processor of an Oracle CEP application.
REPLACERULE	Replaces an existing EPL rule with new EPL text.
GETRULE	Returns the text of an existing EPL rule of the processor of an Oracle CEP application.
UPLOAD	Configures a set of EPL rules for a processor of an Oracle CEP application by uploading the rules from an XML file.
DOWNLOAD	Downloads the set of EPL rules associated with a processor of an Oracle CEP application to a file.
ADDPARAMS	Adds a new set of parameters to a parameterized EPL query.
DELETEPARAMS	Deletes a set of parameters from a parameterized EPL query.
GETPARAMS	Returns the parameters currently bound to a parameterized EPL query.

ADDRULE

Adds a new EPL rule to the specified processor of an Oracle CEP application.

If a rule with the same name (identified with the *rulename* parameter) already exists, then the ADDRULE command replaces the existing rule with the new one.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
    ADDRULE -application application -processor processor -rule [rulename]
rulestring
```

Table 5-7 ADDRULE Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle CEP application whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule [`*rulename*`]` *rulestring*	Specifies the EPL rule you want to add to the specified processor of your application. The *rulename* parameter is not required; if you do not specify it, Oracle CEP generates a name for you. Enter the EPL rule using double quotes.

Example

The following example shows how to add the EPL rule `SELECT * FROM Withdrawal RETAIN 5 EVENTS`, with name `myrule`, to the `helloworldProcessor` of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        ADDRULE -application helloworld -processor helloworldProcessor
        -rule myrule "SELECT * FROM Withdrawal RETAIN 5 EVENTS"
```

DELETERULE

Deletes an existing EPL rule from the specified processor of an Oracle CEP application.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
    DELETERULE -application application -processor processor -rule rulename
```

Table 5-8 DELETERULE Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.

Table 5-8 DELETERULE Arguments

Argument	Definition
`-processor` *`processor`*	Specifies the name of the particular processor, attached to the Oracle CEP application specified with the `-application` argument, whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule` *`rulename`*	Specifies the name of the EPL rule you want to delete. See "Querying for Application and Processor Names" on page 5-35 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 for a particular processor.

Example

The following example shows how to delete the EPL rule called `myrule` from the `helloworldProcessor` of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        DELETERULE -application helloworld -processor helloworldProcessor
        -rule myrule
```

REPLACERULE

Replaces an existing EPL rule with another rule. Oracle CEP first destroys the original rule and then inserts the new one in its place. If the original rule was parameterized, any existing bindings are applied to the new rule.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     REPLACERULE -application application -processor processor -rule
rulename rulestring
```

Table 5-9 REPLACERULE Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule` *`rulename rulestring`*	Specifies the EPL rule you want to replace. Oracle CEP deletes the old rule and then inserts a new one, with the same name but with the new rule text. Enter the EPL rule using double quotes.

Example

The following example shows how to replace a rule called `myrule` with the EPL text `SELECT * FROM Withdrawal RETAIN 10 EVENTS` in the `helloworldProcessor` of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
       REPLACERULE -application helloworld -processor helloworldProcessor
        -rule myrule "SELECT * FROM Withdrawal RETAIN 10 EVENTS"
```

GETRULE

Returns the full text of an EPL rule from the specified processor of an Oracle CEP application.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     GETRULE -application application -processor processor -rule rulename
```

Table 5-10 GETRULE Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule` *`rulename`*	Specifies the name of the EPL rule for which you want to view its full text. See "Querying for Application and Processor Names" on page 5-35 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 for a particular processor.

Example

The following example shows how to get the full text of the EPL rule called `myrule` from the `helloworldProcessor` of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        GETRULE -application helloworld -processor helloworldProcessor
        -rule myrule
```

ADDPARAMS

Adds a new set of parameters to a parameterized EPL query.

See Parameterized Queries for information about using parameterized EPL queries.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     ADDPARAMS -application application -processor processor
     -rule rulename -values values -params params
```

Table 5-11 ADDPARAMS Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle CEP application whose parameterized EPL rule you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose parameterized EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule` *rulename*	Specifies the name of the parameterized EPL rule for which you want add a new set of parameters. See "Querying for Application and Processor Names" on page 5-35 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 for a particular processor.

Table 5-11 ADDPARAMS Arguments

Argument	Definition
`-values values`	Specifies a comma-separated list of values that make up the parameter you want to add. Each value corresponds to a placeholder in the parameterized EPL query.
`-params params`	Specifies a unique identifier for this new parameter set.

Example

The following example shows how to use the ADDPARAMS command:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        ADDPARAMS -application myApplication -processor myProcessor
        -rule MarketRule
        -values "NYSE,BGP" -params nyBGP
```

The example shows how to add a parameter set identified by the string `nyBGP`, with values `NYSE,BGP`, to a parameterized query `MarketRule` running in the `myProcessor` component of `myApplication`. Because the parameter set is composed of two values, the EPL query must contain two placeholders.

DELETEPARAMS

Deletes one or all set of parameters associated with a parameterized EPL query.

See Parameterized Queries for information about using parameterized EPL queries.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     DELETEPARAMS -application application -processor processor
     -rule rulename [-params params]
```

Table 5-12 DELETEPARAMS Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose parameterized EPL rule you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose parameterized EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule` *`rulename`*	Specifies the name of the parameterized EPL rule for which you want to delete one or all of its parameter sets. See "Querying for Application and Processor Names" on page 5-35 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 for a particular processor.
`-params` *`params`*	Specifies the parameter set you want to delete. This argument is optional; if you do not specify it, `wlevs.Admin` deletes all parameter sets currently associated with the parameterized EPL rule.

Example

The following example shows how to use the DELETEPARAMS command:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        DELETEPARAMS -application myApplication -processor myProcessor
        -rule MarketRule
        -params nasORCL
```

The example shows how to delete the parameter set identified with the nasORCL string from the parameterized query `MarketRule` running in the `myProcessor` component of `myApplication`.

To delete all parameter sets associated to the query, do not specify the `-params` option:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        DELETEPARAMS -application myApplication -processor myProcessor
        -rule MarketRule
```

GETPARAMS

Returns one or all the parameter sets currently bound to a parameterized EPL query.

See Parameterized Queries for information about using parameterized EPL queries.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     GETPARAMS -application application -processor processor
     -rule rulename [-params params]
```

Table 5-13 GETPARAMS Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose parameterized EPL rule you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose parameterized EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-rule` *`rulename`*	Specifies the name of the parameterized EPL rule for which you get the parameter sets. See "Querying for Application and Processor Names" on page 5-35 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 for a particular processor.
`-params` *`params`*	Specifies the parameter set you want return. This argument is optional; if you do not specify it, `wlevs.Admin` returns all parameter sets currently associated with the parameterized EPL rule.

Example

The following example shows how to use the GETPARAMS command:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        GETPARAMS -application myApplication -processor myProcessor
         -rule MarketRule
```

The example shows how to get all the parameters currently associated with the parameterized query `MarketRule` running in the `myProcessor` component of `myApplication`. The command would return something like:

```
NASDAQ,ORCL
NYSE,JPM
NYSE,WFC
NYSE,BGP
```

To retrieve a particular parameter set, specify its ID using the `-params` option:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        GETPARAMS -application myApplication -processor myProcessor
        -rule MarketRule
        -params nasORCL
```

UPLOAD

Replaces the configured EPL rules for a specified processor with the EPL rules from an uploaded XML file.

The XML file that contains the list of EPL rules conforms to the processor configuration XSD Schema. This file contains one or more EPL rules that will replace those currently configured for the specified processor. An example of the XML file is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>
  <processor>
    <name>helloworldProcessor</name>
    <rules>
      <rule id="helloworldRule1">
        <![CDATA[  SELECT * FROM HelloWorldEvent RETAIN 2 EVENTS ]]>
      </rule>
    </rules>
  </processor>
</config>
```

In the preceding example, the XML file configures a single rule, with name `helloworldRule1`, and its EPL query text is `SELECT * FROM HelloWorldEvent RETAIN 2 EVENTS`.

WARNING: When you use the UPLOAD command of the `wlevs.Admin` utility, you use the -processor argument to specify the name of the processor to which you want to add the EPL rules, as you do with the other EPL commands. This means that the utility *ignores* any `<name>` elements in the XML file to avoid any naming conflicts.

See Configuring the Complex Event Processor Rules for details and examples of creating the EPL rule XML file.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     UPLOAD -application application -processor processor -sourceURL
sourcefileURL
```

Table 5-14 UPLOAD Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-sourceURL` *`sourcefileURL`*	Specifies the URL of the XML file that contains the EPL rules.

Example

The following example shows how upload the EPL rules in the `c:\processor\config\myrules.xml` file to the `helloworldProcessor` of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        UPLOAD -application helloworld -processor helloworldProcessor
        -sourceURL file:///c:/processor/config/myrules.xml
```

DOWNLOAD

Downloads the set of EPL rules associated with the specified processor of an Oracle CEP application to an XML file.

The XML file is of the same format as described in "UPLOAD" on page 5-25.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     DOWNLOAD -application application -processor processor
     -file destinationfile [-overwrite overwrite]
```

Table 5-15 DOWNLOAD Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 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 CEP application specified with the `-application` argument, whose EPL rules you want to manage. See "Querying for Application and Processor Names" on page 5-35 for details on getting the exact name if you do not know it.
`-file` *`destinationfile`*	Specifies the name of the XML file to which you want the wlevs.Admin utility to download the EPL rules. Be sure you specify the full pathname of the file.
`-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 EPL rules currently attached to the `helloworldProcessor` of the `helloworld` application to the file `c:\processor\config\myrules.xml`; the utility overwrites any existing file:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        DOWNLOAD -application helloworld -processor helloworldProcessor
        -file c:\processor\config\myrules.xml
```

Commands for Managing Oracle CEP MBeans

The following sections describe `wlevs.Admin` commands for managing Oracle CEP MBeans.

See the Javadoc for the full description of the Oracle CEP MBeans.

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 CEP 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 CEP MBean interface classes, see the Javadocs 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.EPLProcessorMBean`, use `EPLProcessor`.

MBean Management Commands

Table 5-16 is an overview of the MBean management commands.

Table 5-16 MBean Management Command Overview

Command	Description
GET	Displays properties of MBeans.
INVOKE	Invokes management operations that an MBean exposes for its underlying resource.

Table 5-16 MBean Management Command Overview (Continued)

Command	Description
QUERY	Searches for MBeans whose `ObjectName` matches a pattern that you specify.
SET	Sets the specified property values for the named MBean instance.

GET

Displays MBean properties (attributes) and JMX object names (in the `javax.management.ObjectName` format).

The output of the command is as follows:

```
{MBeanName object-name {property1 value} {property2 value}. . .}
. . .
```

Note that 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 ]
```

```
GET [-pretty] {-type mbeanType| -mbean objectName} [-property property1]
[-property property2]...
```

Table 5-17 GET Arguments

Argument	Definition
`-type mbeanType`	Returns information for all MBeans of the specified type. For more information, see "Specifying MBean Types" on page 5-29.
`-mbean objectName`	Fully qualified object name of an MBean in the `javax.management.ObjectName` format. For example, if you want to look up an MBean for an EPL Processor Stage, the naming is as follows `"com.bea.wlevs:Name=<name of the Stage>,Type=<type of Mbean>, Application=<name of the application>"`
`-pretty`	Places property-value pairs on separate lines.
`-property property`	The name of the MBean property (attribute) or properties to be listed. **Note:** If property is not specified using this argument, all properties are displayed.

Example

The following example displays all properties of the `EPLProcessorMBean` that was registered for the Processor Stage when the application called `helloworld` was deployed in Oracle CEP.

```
prompt> java wlevs.Admin
         -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
         -username wlevs -password wlevs
         GET -pretty
         -mbean
com.bea.wlevs:Name=eplprocessor,Type=EPLProcessor,Application=helloworld
```

The following example displays all instances of all `EPLProcessorMBean` MBeans.

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        GET -pretty -type EPLProcessor
```

INVOKE

Invokes a management operation for one or more MBeans. For Oracle CEP MBeans, you usually use this command to invoke operations other than the `get`*`Attribute`* and `set`*`Attribute`* that most Oracle CEP MBeans provide.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     INVOKE {-type mbeanType | -mbean objectName} -method methodname
[argument . . .]
```

Table 5-18 INVOKE Arguments

Arguments	Definition
`-type` *`mbeanType`*	Invokes the operation on all MBeans of a specific type. For more information, see "Specifying MBean Types" on page 5-29.
`-mbean` *`objectName`*	Fully qualified object name of an MBean in the `javax.management.ObjectName` format. For example, if you want to invoke an MBean for an EPL 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.
`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 the `com.bea.wlevs.configuration.application.DefaultProcessorConfig` MBean:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        INVOKE -mbean
com.bea.wlevs:Name=eplprocessor,Type=EPLProcessor,Application=helloworld
        -method addRule "SELECT * FROM Withdrawal RETAIN ALL"
```

QUERY

Searches for Oracle CEP MBeans whose `javax.management.ObjectName` matches a pattern that you specify.

All MBeans that are created from an Oracle CEP MBean type are registered in the MBean Server under a name that conforms to the `javax.management.ObjectName` conventions. You must know an MBean's `ObjectName` if you want to use `wlevs.Admin` commands to retrieve or modify specific MBean instances.

The output of the command is as follows:

`{MBeanName` *`object-name`* `{`*`property1 value`*`} {`*`property2 value`*`}. . .}`
`. . .`

Note that 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 5-19 QUERY Arguments

Argument	Definition
`-pretty`	Places property-value pairs on separate lines.
`-pattern` *`object-name-pattern`*	A partial `javax.management.ObjectName` 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 `javax.management.ObjectName`. 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=EPLProcessor,*` 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.configuration.application.DefaultProcessorConfig` MBeans:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
         QUERY -pattern *:Type=EPLProcessor,*
```

If the command succeeds, it returns the following:

```
Ok
```

Querying for Application and Processor Names

All the commands for managing the EPL rules of an Oracle CEP application require you know the name of the application, as well 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 CEP instance:

```
-pattern  com.bea.wlevs:*,Type=EPLProcessor
```

For example:

```
prompt> java wlevs.Admin -url
        service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        QUERY -pretty
        -pattern com.bea.wlevs:*,Type=EPLProcessor
```

A sample output of this command is shown below:

```
Command Output
-------------------------------------------------------
MBeanName:
"com.bea.wlevs:Name=helloworldProcessor,Type=EPLProcessor,Application=hellowor
ld,"
       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`.

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 5-20 SET Arguments

Argument	Definition
`-type` *`mbeanType`*	Sets the properties for all MBeans of a specific type. For more information, see "Specifying MBean Types" on page 5-29.
`-mbean` *`objectName`*	Fully qualified object name of an MBean in the `javax.management.ObjectName` format: `"com.bea.wlevs:Name=<name of the stage>,Type=<MBean type>,Application=<name of the deployed application>"`

Table 5-20 SET Arguments

Argument	Definition
`-property` *property*	The name of the property to be set.

property _value	The value to be set.
	• Some properties require you to specify the name of an Oracle CEP MBean. In this case, specify the fully qualified object name of an MBean in the `javax.management.ObjectName` format. For example: `"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 stream named `helloworldOutstream` of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        SET -mbean
com.bea.wlevs:Name=helloworldOutstream,Type=Stream,Application=helloworld
        -property  MaxSize 1024
```

Commands for Controlling Event Record and Playback

Table 5-21 is an overview of commands for managing event record and playback for a particular stage of an Oracle CEP application. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table 5-21 Overview of Commands for Controlling Event Record and Playback

Command	Description
STARTRECORD	Starts the recording of events for a stage in an Oracle CEP application.
ENABLEPLAYBACK	Enables the playback of events for a stage in an Oracle CEP application.
STOPRECORD	Stops the recording of events for a stage in an Oracle CEP application.
DISABLEPLAYBACK	Disables the playback of events for a stage in an Oracle CEP application.
CONFIGURERECORD	Configures the parameters for the event recording of a stage in an Oracle CEP application.
CONFIGUREPLAYBACK	Configures the parameters for the event playback of a stage in an Oracle CEP application.

STARTRECORD

Starts the recording of events for any particular stage of an Oracle CEP application.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     STARTRECORD -application application -stage stage
```

Table 5-22 STARTRECORD Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose event record and playback you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle CEP 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 CEP instance:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        STARTRECORD -application helloworld -stage helloworldAdapter
```

ENABLEPLAYBACK

Enables the playing back of events of a particular stage of a Oracle CEP application.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     ENABLEPLAYBACK -application application -stage stage
```

Table 5-23 ENABLEPLAYBACK Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose event record and playback you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle CEP application specified with the `-application` argument, whose event record and playback you want to manage.

Example

The following example shows how to enable the playback of events on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle CEP instance:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        ENABLEPLAYBACK -application helloworld -stage helloworldAdapter
```

STOPRECORD

Stops the recording of events for a stage of an Oracle CEP 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 5-24 STOPRECORD Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose event record and playback you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle CEP 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 CEP instance; it is assumed that the recording of events was previously started for the stage:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        STOPRECORD -application helloworld -stage helloworldAdapter
```

DISABLEPLAYBACK

Disables the playback of events for a stage of an Oracle CEP application in which the playback of events has been previously enabled.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     DISABLEPLAYBACK -application application -stage stage
```

Table 5-25 DISABLEPLAYBACK Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose event record and playback you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle CEP application specified with the `-application` argument, whose event record and playback you want to manage.

Example

The following example shows how to disable the playback of events on the `helloworldAdapter` stage of the `helloworld` application deployed to the specified Oracle CEP instance; it is assumed that the playback of events was previously enabled for the stage:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        DISABLEPLAYBACK -application helloworld -stage helloworldAdapter
```

CONFIGURERECORD

Configures the parameters associated with the recording of events for a stage of an Oracle CEP application.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
```

```
CONFIGURERECORD -application application -stage stage
-startTime startTime [-endTime endTime] [-duration duration]
```

Table 5-26 CONFIGURERECORD Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose event record and playback you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle CEP application specified with the `-application` argument, whose event record and playback you want to manage.
`-startTime` *`startTime`*	Specifies the time when the recording should start. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.
`-endTime` *`endTime`*	Specifies the actual time when the recording should end. Specify `null` if you want the recording to run forever. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`. **Note:** You can specify either `-endTime` or `-duration`, but not both.
`-duration` *`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`. **Note:** You can specify either `-endTime` or `-duration`, 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 CEP instance.

The following example specifies a start and end time for recording:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        CONFIGURERECORD -application helloworld -stage helloworldAdapter
        -startTime 10-20-2007:11:22:07 -endTime 10-20-2007:12:22:07
```

The following example specifies a start and a duration for recording:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        CONFIGURERECORD -application helloworld -stage helloworldAdapter
        -startTime 10-20-2007:11:22:07 -duration 01:00:00
```

The following example specifies a start and a duration of `null`, which means recording with run forever:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        CONFIGURERECORD -application helloworld -stage helloworldAdapter
        -startTime 10-20-2007:11:22:07 -duration null
```

CONFIGUREPLAYBACK

Configures the parameters associated with the playback of events for a stage of an Oracle CEP application.

Syntax

```
java wlevs.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     [ Common Arguments ]
     CONFIGUREPLAYBACK -application application -stage stage
     -startTime startTime [-endTime endTime] [-duration duration]
     [-speed speed] [-loopback loopback]
```

Table 5-27 CONFIGUREPLAYBACK Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose event record and playback you want to manage. See "Querying for Application and Processor Names" on page 5-35 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.
`-stage` *`stage`*	Specifies the name of the particular stage, attached to the Oracle CEP application specified with the `-application` argument, whose event record and playback you want to manage.
`-startTime` *`startTime`*	Specifies the time when the playback should start. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`.
`-endTime` *`endTime`*	Specifies the actual time when the playback should end. Specify `null` if you want the playback to run forever. The format is `MM-dd-yyyy:HH:mm:ss`, such as `10-20-2007:11:22:07`. **Note:** You can specify either `-endTime` or `-duration`, but not both.
`-duration` *`duration`*	Specifies the duration of time after which event playback for this stage ends. Specify `null` if you want the playback to run forever. The format is `HH:mm:ss`, such as `01:00:00`. **Note:** You can specify either `-endTime` or `-duration`, but not both.

Table 5-27 CONFIGUREPLAYBACK Arguments

Argument	Definition
`-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.
`-loopback` *loopback*	Specifies whether to playback events again after the playback of the specified time interval is over. Valid values are `true` and `false`. Default value is `true`. A value of `true` means that the repeat of playback continues an infinite number of times until it is deliberately stopped. `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 CEP 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:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
       CONFIGUREPLAYBACK -application helloworld -stage helloworldAdapter
        -startTime 10-20-2007:11:22:07 -endTime 10-20-2007:12:22:07
        -speed 2 -loopback 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:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
       CONFIGUREPLAYBACK -application helloworld -stage helloworldAdapter
        -startTime 10-20-2007:11:22:07 -duration 01:00:00
        -speed 0.5 -loopback false
```

The following example specifies a start and a duration of `null`, which means playback with run forever at normal speed with loopback:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
       CONFIGUREPLAYBACK -application helloworld -stage helloworldAdapter
        -startTime 10-20-2007:11:22:07 -duration null
```

Commands for Monitoring Throughput and Latency

Table 5-21 is an overview of commands for monitoring throughput and latency in an Oracle CEP application. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table 5-28 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.*
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.

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 5-29 MONITORAVGLATENCY Arguments

Argument	Definition
`-application` *application*	Specifies the name of the Oracle CEP application whose latency you want to monitor. See "Querying for Application and Processor Names" on page 5-35 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.
`-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`.
`-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`.

Table 5-29 MONITORAVGLATENCY Arguments

Argument	Definition
`-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.
`-timeUnit` *`timeUnit`*	Specifies the time unit for the latency calculation. Valid values are `MICROSECONDS`, `MILLISECONDS`, and `SECONDS`. Default value is `MILLISECONDS`.

Example

The following example shows how to monitor the average latency of events flowing through the `eplprocessor` 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 (`eplprocessor`), the latency monitoring is happening for just the events flowing through a single stage:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        MONITORAVGLATENCY -application helloworld -startStage eplprocessor
        -startStagePoint entry -endStage eplprocessor -endStagePoint exit
        -avgInterval 100 -timeUnit MILLISECONDS
```

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 5-30 MONITORAVGLATENCYTHRESHOLD Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose latency threshold you want to monitor. See "Querying for Application and Processor Names" on page 5-35 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.
`-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.

Table 5-30 MONITORAVGLATENCYTHRESHOLD Arguments

Argument	Definition
`-timeUnit` *`timeUnit`*	Specifies the time unit for the latency threshold calculation. Valid values are `MICROSECONDS`, `MILLISECONDS`, and `SECONDS`. Default value 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 `eplprocessor` stage, from entry point to exit point, of the `helloworld` application.

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        MONITORAVGLATENCY -application helloworld -startStage eplprocessor
        -startStagePoint entry -endStage eplprocessor -endStagePoint exit
        -avgInterval 100 -timeUnit MILLISECONDS -threshold 100
```

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 5-31 MONITORMAXLATENCY Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose maximum latency you want to monitor. See "Querying for Application and Processor Names" on page 5-35 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.
`-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`.
`-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.
`-timeUnit` *`timeUnit`*	Specifies the time unit for the maximum calculation. Valid values are `MICROSECONDS`, `MILLISECONDS`, and `SECONDS`. Default value is `MILLISECONDS`.

Example

The following example shows how to monitor the maximum latency of events flowing through the `eplprocessor` stage, from entry point to exit point, of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
        MONITORMAXLATENCY -application helloworld -startStage eplprocessor
        -startStagePoint entry -endStage eplprocessor -endStagePoint exit
        -maxInterval 100 -timeUnit MILLISECONDS
```

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 5-32 MONITORAVGLATENCY Arguments

Argument	Definition
`-application` *`application`*	Specifies the name of the Oracle CEP application whose throughput you want to monitor. See "Querying for Application and Processor Names" on page 5-35 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.
`-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.
`-timeUnit` *`timeUnit`*	Specifies the time unit for the throughput calculation. Valid values are `MICROSECONDS`, `MILLISECONDS`, and `SECONDS`. Default value is `MILLISECONDS`.

Example

The following example shows how to monitor the number of events flowing through the entry point of the `eplprocessor` stage of the `helloworld` application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:9004/jmxrmi
        -username wlevs -password wlevs
```

```
MONITORMAXLATENCY -application helloworld
-stage eplprocessor -stagePoint entry
-throughputInterval 100 -avgInterval 100 -timeUnit MILLISECONDS
```

CHAPTER 6

Managing Applications, Servers, and Domains Using MBeans

This section contains information on the following subjects:

Overview of Management

Oracle CEP applications define an event processing network (EPN) that is made up of components such as processors, streams, and adapters. You deploy these applications to an Oracle CEP instance that has been started in a domain.

Note: Components are also sometimes referred to as *stages*, in particular in the management Javadocs. However, for consistency with the rest of the Oracle CEP documentation, this section uses the term *components*.

You can dynamically configure each component in the EPN using managed beans, or *MBeans*. Typical configuration tasks include adding and removing EPL rules, changing stream max size, subscribing to notifications, and executing operations. You manipulate the MBeans either by using the standard Java Management Extension (JMX) APIs, using `wlevs.Admin` (the Oracle

CEP administration command-line utility) or Visualizer, a graphical administration tool. It is assumed in this section you are going to use JMX. See the following sections for information on using `wlevs.Admin` or Visualizer:

- "wlevs.Admin Command-Line Reference" on page 5-1
- Visualizer Help

You can also perform some configuration and application life cycle management of the server, domain, and deployed applications using MBeans, although this section predominantly describes configuring individual application components. However, because server, domain, and application configuration is also done using MBeans, much of the information in this section is applicable.

Each component in a deployed application (adapter, stream, 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 stream or the EPL rules associated with a processor.

You can also gather monitoring information for each component in the EPN using *runtime 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)

Overview of Oracle CEP MBeans

Oracle CEP exposes the following types of MBeans:

- **Configuration MBeans**—Contain information about the configuration of components in an EPN, a deployed Oracle CEP application, the server and domain configurations. These MBeans have a fixed management interface and represent the information contained in the domain config.xml file and the component configuration XML files. Examples of standard MBeans include `EPLProcessorMBean` and `StreamMBean`.

  See "Configuration MBeans" on page 6-3 for additional information.

- **Runtime MBeans**—Contain information about throughput and latency of a component.

  See "Runtime MBeans" on page 6-3.

For full reference information about Oracle CEP MBeans and management in general, see the following Javadocs:

- `com.bea.wlevs.management.configuration`
- `com.bea.wlevs.management.runtime`

- `com.bea.wlevs.monitor.management`
- `com.bea.wlevs.monitor`
- `com.bea.wlevs.processor.epl.management`
- `com.bea.wlevs.deployment.mbean`

Configuration MBeans

When you deploy an Oracle CEP application, the server automatically creates a configuration MBean for each component in the EPN whose manageability has been enabled, or in other words, for each component registered in the EPN assembly file. If you have extended the configuration of an adapter, then the server deploys a a custom configuration MBean for the adapter.

Using JMX, you can dynamically configure the component using its configuration MBean. For example, using the `StreamMBean.setMaxSize()` method you can set the size of a stream component.

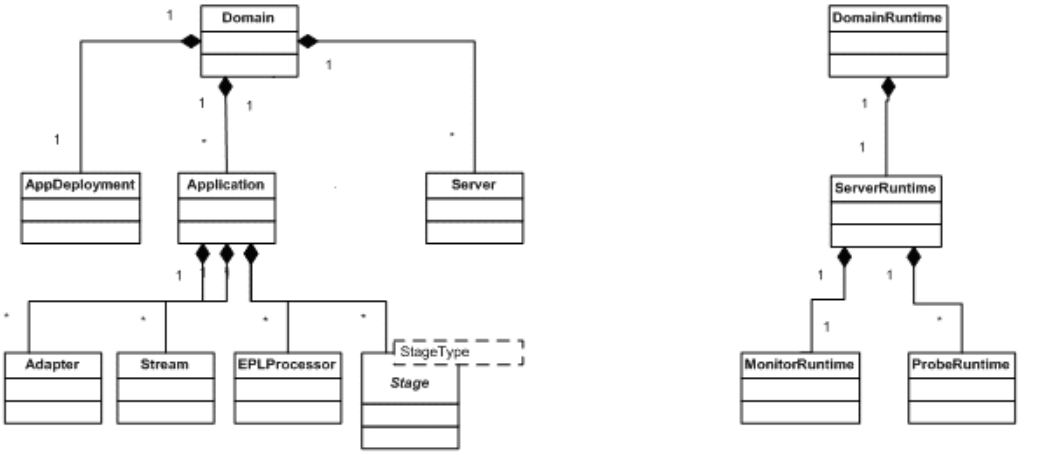
Runtime MBeans

You can also gather monitoring information for each component in the EPN using *runtime MBeans*. WebLogic Event 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 averge 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.

MBean Hierarchy

The following graphic describes the Oracle CEP MBean tree.

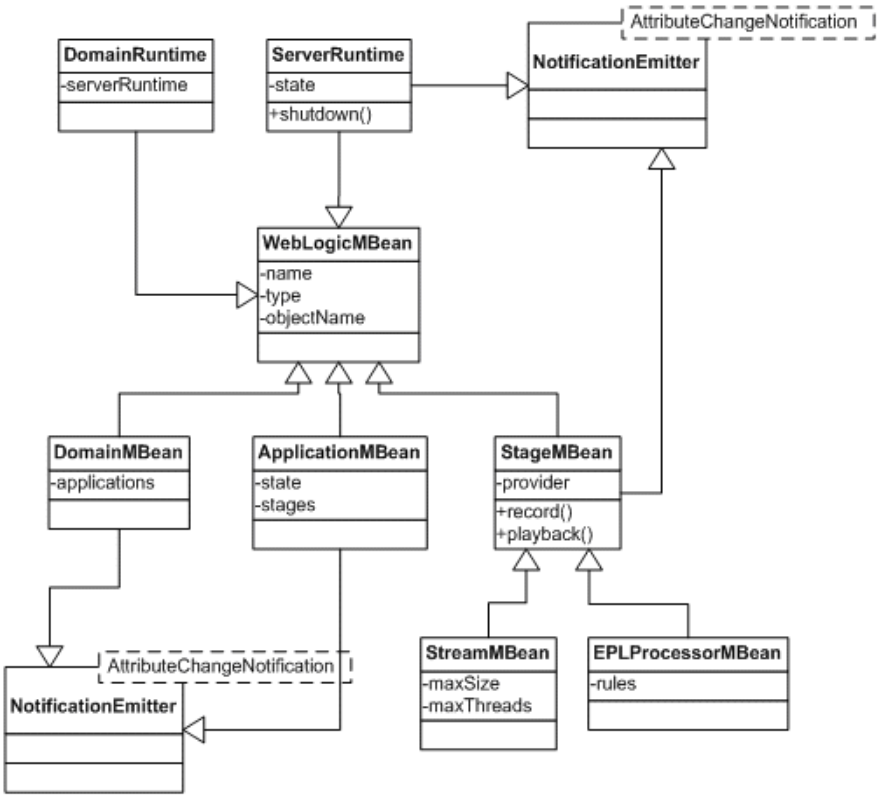


All MBeans must be registered in an MBean server under an object name of type `javax.management.ObjectName`. Oracle CEP follows a convention in which object names for child MBeans contain part of its parent MBean object name.

There are two main MBean roots: `DomainMBean` and `DomainRuntimeMBean`. The former includes configuration MBeans for the entire domain, the latter contains runtime information, such as statistics, and local services, such as Monitor, that are generally scoped 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.

The following diagram shows the main classes and relationships that make up the object model.



Most MBeans are notification emitters that generate `AttributeChangeNotifications`. In other words, a JMX client can register to receive attribute change notifications regarding changes to application state, insertion and removal of applications at the domain, stream size and thread changes, insertion and removal of rules, and so on.

MBean Naming

This section is divided into configuration and runtime MBean naming.

Configuration MBean Naming

Oracle CEP 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=EPLProcessor,Application=myapplication
```

The following table describes the key properties that Oracle WebLogic Server encodes in its MBean object names.

Table 6-1 Oracle CEP 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" manageable="true" />` In this case, the key property would be `Name=myprocessor`.

Table 6-1 Oracle CEP MBean Object Name Key Properties

This Key Property	Specifies
`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 `EPLProcessorMBean`, use `EPLProcessor`. In this case, the key property would be `Type=EPLProcessor`.

`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 CEP 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`.

The following table shows examples of configuration MBean objects 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 6-2 Component Declaration Example With Corresponding MBean Object Names

Sample Component Declaration in EPN Assembly File	Corresponding Configuration MBean Object Name
`<wlevs:processor id="helloworldProcessor" />`	`com.bea.wlevs:Name=helloworldProcessor, Type=EPLProcessor,Application=helloworld,Domain=mydomain` `EPLProcessor` is the standard configuration MBean for processor components. The manageable property is `rules`.

`<wlevs:stream id="helloworldInstream">`
`<wlevs:listener ref="helloworldProcessor"/>`
`<wlevs:source ref="helloworldAdapter"/>`
`</wlevs:stream>`
	`com.bea.wlevs:Name=helloworldInstream,Type=Stream,Application=helloworld,Domain=mydomain`
	`Stream` is the standard configuration MBean for a stream component. The manageable properties are `MaxSize` and `MaxThreads`.

Runtime MBean Naming

Runtime MBeans are named using the same pattern as with configuration mbeans except for one extra property: `Direction`. This property has two valid values: `OUTBOUND` or `INBOUND` that refere 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; similary `INBOUND` means you want to gather the monitoring information as events flow into a component.

For example, the object name of the runtime 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=EPLProcessor,Application=myapplication,Direction=INBOUND
```

See "Configuration MBean Naming" on page 6-5 for details about configuration MBean naming.

Dynamically Configuring a Component Using JMX: Typical Steps

It is assumed in this section that you are going to use the Java Management Extensions (JMX) APIs to manipulate the configuration MBeans. If you want to use `wlevs.Admin`, see Chapter 5, "wlevs.Admin Command-Line Reference." Be sure you have read the following sections that describe Oracle CEP configuration MBeans:

- "Overview of Management" on page 6-1
- "Overview of Oracle CEP MBeans" on page 6-2
- "MBean Hierarchy" on page 6-4
- "MBean Naming" on page 6-5

To dynamically configure a component of an EPN, follow these steps:

1. Be sure that the JMX service is configured for your domain. For details see Chapter 9, "Configuring JMX for Oracle Complex Event Processing."
2. Write the JMX Java code to configure the component using the appropriate MBean. See "Programming with JMX" on page 6-9 for some programming hints.

Programming with JMX

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 CEP server. The following code snippet shows an example:

```
public static void initConnection(String hostname, int port, String username,
char[] password)
  throws IOException,MalformedURLException {

  JMXServiceURL serviceURL = new JMXServiceURL("rmi", "localhost", 9004,
"/jndi/rmi://localhost:" + 9004 + "/jmxrmi");
  System.out.println("Service: " + serviceURL.toString());
  Map<String,Object> h = makeSecureEnv();
  connector = JMXConnectorFactory.connect(serviceURL, h);
  connection = connector.getMBeanServerConnection();
}
```

The `JMXConnectorFactory.connect()` method takes as a second parameter 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;

}
```

The example then shows how 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 ;
}
```

Dynamically Monitoring the Throughput and Latency of a Component

The first thing you do is get an instance of a `MonitorRuntimeMBean` for the component you want to monitor. Be sure you specify whether you want to monitor incoming events (`INBOUND`) or outgoing events (`OUTBOUND`). For example:

```
  m_processorInbound =

ObjectName.getInstance("com.bea.wlevs:Name=myprocessor,Type=EPLProcessor,Appli
cation=myapplication,Direction=INBOUND");
```

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 retuan `ProbeRuntimeMBean`. For example:

```
ObjectName probeON =
m_testBean.getMonitorRuntimeMBean().monitorAvgThroughput(m_processorInbound,
1000, 1000);
```

Once you you have an instance of the `ProbeRuntimeMbean`, you have two ways of getting the actual runtime metrics:

- Use the `getMetric()` method of `ProbeRuntimeMBean` to pull the information.
- Use `javax.management.NotificationBroadcaster.addNotificationListener()` to have the information pushed to you every time there is a change in the metrics.

When you are finished gathering monitoring information, use `ProbeRuntimeMBean.terminate()` to unregister the MBean from the MBean server.

For additional details about these MBean interfaces and how to use them to monitor throughput and latency, see runtime monitoring Javadocs.

CHAPTER 7

Configuring Security for Oracle CEP

This section contains information on the following subjects:

Overview of Security in Oracle CEP

Oracle Complex Event Processing, or *Oracle CEP* for short, provides a variety of mechanisms to protect server resources such as data and event streams, configuration, username and password data, security policy information, remote credentials, and network traffic.

Oracle CEP supports various security providers for authentication, authorization, role mapping, and credential mapping. As initially installed, Oracle CEP is configured to use the file-based providers for both authentication and authorization. You can also configure the system to use an LDAP or DBMS provider.

Oracle CEP uses role-based authorization control to secure the Visualizer Administration Console and the `wlevs.Admin` command-line utility. There are six default out-of-the-box security groups. You can add users to different groups to give them the different roles. Oracle CEP also provides one-way SSL to protect network traffic between Visualizer and the server instance upon which the data-services application runs, as well as the network traffic between server instances of a multi-server domain.

Security Providers

Oracle CEP supports the following security providers:

- File-based—Default out-of-the-box security provider. This type of provider uses an operating system file to access security data such as user, password, and group information. Provides both authentication (process whereby identity of users is proved or verified) and authorization (process whereby a user's access to an Oracle CEP resource is permitted or denied based on the user's security role and the security policy assigned to the requested Oracle CEP resource). Authentication typically involves username/password combinations.
- LDAP—Provider that uses a Lightweight Data Access Protocol (LDAP) server to access user, password, and group information. Provides only authentication.
- DBMS—Provider that uses a database management system (DBMS) to access user, password, and group information. Provides both authentication and authorization.

If you choose to use the default file-based security provider, then you do not need to do any further configuration of your domain because the Configuration Wizard did it for you. However, if you want to use the LDAP or DBMS providers, further configuration is required.

Because the LDAP provider can be used only for authentication, while the DBMS provider can be used for both authentication and authorization, the following configurations are discussed in this section:

- "Using the LDAP Provider For Authentication and DBMS Provider for Authorization" on page 7-5
- "Using the DBMS Provider for Both Authentication and Authorization" on page 7-11

Once you have configured the security provider, you can start using Visualizer to add new users, assign them to groups, and map groups to roles. See "Overview of Users, Groups, and Roles" on page 7-3 for general information, and then Overview of Visualizer for instructions on using Visualizer.

Overview of Users, Groups, and Roles

Administrators who use Visualizer, `wlevs.Admin`, or any custom administration application that uses JMX to connect to an Oracle CEP instance use role-based authorization to gain access. Users that successfully authenticate themselves when using 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 Visualizer or `wlevs.Admin`.

When an administrator uses the Configuration Wizard to create a new domain, they enter an administrator user that will be 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. Once the domain has been created, the administrator can create new groups using Visualizer, assign roles to them, and then create new users and assign them to groups.

The following table describes the default Oracle CEP security 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 7-1 Available Oracle CEP Roles and Groups

Role	Description	Associated Group Name
Operator	Has read-only access to all server resources, services, and deployed applications.	`wlevsOperators`
Monitor	Has all Operator privileges as well as permission to enable/disable diagnostic functions, such as creating a diagnostic profile and recording events (then playing them back.)	`wlevsMonitors`
ApplicationAdmin	Has all Operator privileges as well as permission to update the configuration of any deployed application.	`wlevsApplicationAdmins`

Table 7-1 Available Oracle CEP Roles and Groups

Role	Description	Associated Group Name
Deployer	Has all Operator privileges as well as permission to deploy, undeploy, update, suspend, and resume any deployed application.	`wlevsDeployers`
BusinessUser	Has all Operator privileges as well as permission to update the EPL rules associated with the processor of a deployed application.	`wlevsBusinessUsers`
Admin	Has all privileges of all the preceding roles, as well as permission to: • Create users and groups • Configure HTTP publish-subscribe security • Change the system configuration, such as Jetty, work manager, and so on.	`wlevsAdministrators`

Security in Oracle CEP 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 keystore. This user is automatically added to the `wlevsAdministrators` group. All security configuration is stored using a file-based provider, by default.

All Oracle CEP examples are configured to have an administrator with username `wlevs` and password `wlevs`. 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 `wlevs`, with password `wlevs`, is configured to be the Oracle CEP administrator with full administrator privileges. The scripts to start the server for these examples use the appropriate arguments to pass this username and password to the `java` command. If you use the Deployer or `wlevs.Admin` utility, you must also pass this username/password pair using the appropriate arguments.

Securely Specifying User Credentials When Using the Command-Line Utilities

Oracle CEP includes the following command-line utilities for performing a variety of tasks:

- `wlevs.Admin`. See "wlevs.Admin Command-Line Reference" on page 5-1 for details
- `Deployer`. See Deployer Command-Line Reference for details.
- `cssconfig`. See "The cssconfig Command Line Utility" on page 7-26 for details.
- `encryptMSAConfig`. See "The encryptMSAConfig Command Line Utility" on page 7-27 for details.

For each utility, you can specify user credentials (username 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 CEP includes a set of standard native libraries for this purpose, but it may not include the specific one you need.

Using the LDAP Provider For Authentication and DBMS Provider for Authorization

The following procedure describes how to configure the LDAP security provider for authentication and the DBMS provider for authorization.

WARNING: When using LDAP for authentication, you can not add or delete users and groups using Visualizer, you can only change the password of a user.

1. Open a command window and set your environment as described in Setting Up Your Development Environment.

2. Add the *ORACLE_CEP_HOME*\ocep_10.3\bin directory to your PATH environment variable, where *ORACLE_CEP_HOME* is the main Oracle CEP installation directory, such as d:\oracle_cep:

   ```
   prompt> set PATH=d:\oracle_cep\ocep_10.3\bin;%PATH% (Windows)
   ```

   ```
   prompt> PATH=/oracle_cep/ocep_10.3/bin:$PATH (UNIX)
   ```

3. Change to the *DOMAIN_DIR*/*servername*/config directory, where *DOMAIN_DIR* refers to the main directory of your domain, such as d:\oracle_cep\user_projects\domains\mydomain, and *servername* refers to the name of your server:

   ```
   prompt> cd
   d:\oracle_cep\user_projects\domains\mydomain\defaultserver\config
   ```

4. Using your favorite text editor, create a file called myLDAPandDBMS.properties and copy into it the entire contents of the section "Sample LDAP/DBMS Properties File" on page 7-7.

   Customize the property file by updating the store.StoreProperties property to reflect your database driver information, connection URL, and username 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
   ```

   Also update the property that specifies your LDAP server configuration.

   Leave all the other properties to their default values.

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

   ```
   prompt> copy security.xml security.xml_save
   ```

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

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

   In the preceding command, myLDAPandDBMS.properties is the property file you created in step 4, security.xml is the name of the new security configuration file, and security-key.dat is an existing file, generated by the Configuration Wizard, that contains the identity key.

   See "The cssconfig Command Line Utility" on page 7-26 for additional information.

7. Change to the *ORACLE_CEP_HOME*/ocep_10.3/utils/security/sql directory:

```
prompt> cd d:\oracle_cep\ocep_10.3\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.

8. Run the following SQL script against the database you specified as the database store in step 4:
   - `atz_create.sql`
9. Configure your LDAP server by adding the default groups described in "Overview of Users, Groups, and Roles" on page 7-3 as well as the administrator user you specified when you created the domain. By default, this user is called `wlevs`.

   Refer to your LDAP server documentation for details.

Sample LDAP/DBMS Properties 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:
#  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.LDAPSSLSocketFactoryLoo
kupServiceMBean
#ldapssl.Protocol=
#ldapssl.TrustManagerClassName=

namedsql.mbean=com.bea.common.management.configuration.NamedSQLConnectionLooku
pServiceMBean

store.mbean=com.bea.common.management.configuration.StoreServiceMBean
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.OpenLDAPAuthenticatorM
Bean
#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.DefaultCredentialMapperMBea
n
cm.1.Name=my-credential-mapper
cm.1.CredentialMappingDeploymentEnabled=true

#cm.3.mbean=weblogic.security.providers.credentials.FileBasedCredentialMapperM
Bean
#cm.3.FileStorePath=
#cm.3.FileStorePassword=
#cm.3.EncryptAlgorithm=
#cm.3.Name=
#cm.3.CredentialMappingDeploymentEnabled=

rm.1.mbean=weblogic.security.providers.xacml.authorization.XACMLRoleMapperMBea
n
rm.1.Name=my-role-mapper
rm.1.RoleDeploymentEnabled=true
```

```
atz.1.mbean=weblogic.security.providers.xacml.authorization.XACMLAuthorizerMBe
an
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
```

Using the DBMS Provider for Both Authentication and Authorization

The following procedure describes how to configure the DBMS security provider for both authentication and authorization.

1. Open a command window and set your environment as described in Setting Up Your Development Environment.
2. Add the *ORACLE_CEP_HOME*`\ocep_10.3\bin` directory to your `PATH` environment variable, where *ORACLE_CEP_HOME* is the main Oracle CEP installation directory, such as `d:\oracle_cep`:

   ```
   prompt> set PATH=d:\oracle_cep\ocep_10.3\bin;%PATH% (Windows)

   prompt> PATH=/oracle_cep/ocep_10.3/bin:$PATH (UNIX)
   ```

3. Change to the *DOMAIN_DIR/servername*`/config` directory, where *DOMAIN_DIR* refers to the main directory of your domain, such as `d:\oracle_cep\user_projects\domains\mydomain`, and *servername* refers to the name of your server:

   ```
   prompt> cd
   d:\oracle_cep\user_projects\domains\mydomain\defaultserver\config
   ```

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

   ```
   prompt> copy security.xml security.xml_save
   ```

5. Using your favorite text editor, create a file called `myDBMS.properties` and copy into it the entire contents of the section "Sample DBMS Property File" on page 7-13.

   Customize the property file by updating the `store.StoreProperties` property to reflect your database driver information, connection URL, and username 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
```

Leave all the other properties to their default values.

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

   ```
   prompt> cssconfig -p myDBMS.properties -c security.xml -i
   security-key.dat
   ```

   In the preceding command, `myDBMS.properties` is the property file you created in step 4, `security.xml` is the name of the new security configuration file, and `security-key.dat` is an existing file, generated by the Configuration Wizard, that contains the identity key.

   See "The cssconfig Command Line Utility" on page 7-26 for additional information.

7. Change to the *`ORACLE_CEP_HOME`*`/ocep_10.3/utils/security/sql` directory:

   ```
   prompt> cd d:\oracle_cep\ocep_10.3\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 `wlevs`, with password `wlevs`, into the user table and specifies that the user belongs to the wlevsAdministrators group. The script also insers the default groups listed in Table 7-1 into the group table.
   - `atz_create.sql`—Creates all tables required for authorization.
   - `atz_drop.sql`—Drops all authorization-related tables.

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

   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');
```

9. Run the following SQL script files, in the order listed, against the database you specified as the database store in step 4:
   - atn_create.sql
   - atn_init.sql
   - atz_create.sql

Sample DBMS Property 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:
#  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.LDAPSSLSocketFactoryLoo
kupServiceMBean
#ldapssl.Protocol=
#ldapssl.TrustManagerClassName=

namedsql.mbean=com.bea.common.management.configuration.NamedSQLConnectionLooku
pServiceMBean
```

```
store.mbean=com.bea.common.management.configuration.StoreServiceMBean
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.NamedSQLConnectionMBea
n
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.DefaultCredentialMapperMBea
n
cm.1.Name=my-credential-mapper
cm.1.CredentialMappingDeploymentEnabled=true

rm.1.mbean=weblogic.security.providers.xacml.authorization.XACMLRoleMapperMBea
n
```

```
rm.1.Name=my-role-mapper
rm.1.RoleDeploymentEnabled=true

atz.1.mbean=weblogic.security.providers.xacml.authorization.XACMLAuthorizerMBe
an
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
```

Configuring Password Strength

Password strength is a measurement of the effectiveness of a password as an authentication credential. How the password strength is configured determines the type of password a user can specify, such as whether the password can contain the username, 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 CEP authentication by updating the security configuration file (`security.xml`), located in the *`DOMAIN_DIR`*`/`*`servername`*`/config` directory, where *`DOMAIN_DIR`* refers to your domain directory, such as `d:/oracle_cep/user_projects/domains/mydomain`, and *`servername`* refers to your server, such as `defaultserver`.

The password strength configuration is contained in the `<password-validator>` element. The following snippet from the `security.xml` file shows the default values after creating a new domain using the Configuration Wizard:

```
<sec:password-validator

xmlns:pas="http://www.bea.com/ns/weblogic/90/security/providers/passwordvalida
tor"
   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-userna
me>

<pas:reject-equal-or-contain-reverse-username>false</pas:reject-equal-or-conta
in-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>
```

The following table describes all the child elements of `<password-validator>` you can configure and what they mean.

If you manually update the `security.xml` file, you must restart the Oracle CEP server instance for the changes to take effect.

Table 7-2

Child Element of `<password-validator>`	Description	Default Value
reject-equal-or-contain-name	When set to `true`, Oracle CEP rejects a password if it is the same as, or contains, the username. When set to `false`, Oracle CEP does not reject a password for this reason.	true
reject-equal-or-contain-reverse-username	When set to `true`, Oracle CEP rejects a password if it is the same as, or contains, the reversed username. When set to `false`, Oracle CEP 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

Table 7-2

Child Element of `<password-validator>`	Description	Default Value
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
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

Table 7-2

Child Element of <password-validator>	Description	Default Value
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

Changing the Default Administration User

When you create a server using the Configuration Wizard, you enter the name and password of the main administrative user. If you later need to change this user, follow these steps:

1. Using Visualizer, create a new user and assign it to the `wlevsAdministrators` group. Be sure that the `wlevsAdministrators` group is mapped to the `Admin` role. See Typical Security Tasks in the Visualizer online help for details.

   For simplicity, assume that the new user is called `cepAdmin`, with password `supersecret`.

2. Update the *`DOMAIN_DIR`*`/`*`servername`*`/config/security-config.xml` file of the server, changing the values of the `<boot-user-name-encrypted>` and `<password>` child elements of `<msa-security>` accordingly, as shown below:

   ```
   <msa-security>
      <boot-user-name-encrypted>cepAdmin</boot-user-name-encrypted>
      <password>supersecret</password>
   </msa-security>
   ```

3. Encrypt the cleartext password in the `<password>` of the `security-config.xml` file by using the encryption utility. See "The encryptMSAConfig Command Line Utility" on page 7-27.

Using SSL to Secure Network Traffic

Oracle CEP uses 1-way SSL to secure the network traffic between:

- A browser running the Visualizer Administration Console and the Oracle CEP instance that hosts the data-services application used by Visualizer.
- The `wlevs.Admin` command-line utility and an Oracle CEP instance.
- The member servers of a multi-server domain.

How SSL Is Configured in Oracle CEP

This section how SSL is configured in Oracle CEP, as well as the default SSL configuration for a new server.

SSL is configured in the server's `config.xml` file. When you create an Oracle CEP server using the Configuration Wizard, the server's `config.xml` automatically includes a default SSL configuration. In particular, the `config.xml` file includes a `<netio>` configuration object that specifies the secure port (9003 by default); for example:

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

The Jetty HTTP server's configuration in turn configures this network i/o port for its secure connection:

```
<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 port is configured at the time the Configuration Wizard is run, but you can later change it to suit your needs.

The SSL configuration itself looks like the following:

```
<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 *`DOMAIN_DIR/servername/`*`ssl` directory; its password is the same as that entered when creating a server with the Configuration Wizard. The `evsidentity.jks` contains a self-signed certificate. You can create your own certificate file and either replace the `evsidentity.jks` file, or update the `<key-store>` element in the `config.xml` file.

Configuring SSL In a Multi-Server Domain for Use By Visualizer

The following procedure shows how to configure one-way SSL between the server that hosts the Visualizer data-services application and another server in a multi-server domain.

In the procedure, it is assumed that the server that hosts the Visualizer's data-services application is called `server1` and the other server is called `server2`, and that both are located in the `/oracle_cep/user_projects/domains/mydomain` directory. Repeat this procedure for other servers in the domain, if required.

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 "How SSL Is Configured in Oracle CEP" on page 7-20 for details, as well as information on how to change the default configuration.
2. Start `server2`.
3. Change to the `ssl` sub-directory of the main `server1` directory:

   ```
   prompt> cd /oracle_cep/user_projects/domains/mydomain/server1/ssl
   ```

4. Generate a trust keystore for `server1` (that includes the certificate of `server2`) by specifying the following command:

```
prompt> java GrabCert host:secureport [-alias=alias] [-noinput]
[truststorepath]
```

where

- *host* refers to the computer on which `server2` is running.
- *secureport* refers to the SSL network i/o port configured for `server2`; see "How SSL Is Configured in Oracle CEP" on page 7-20. Default value is 4098
- *alias* refers to the alias for the certificate in the trust keystore. Default value is the hostname.
- *truststorepath* refers to the full pathname of the generated trust keystore file; default is `evstrust.jks`

For example:

```
prompt> java GrabCert ariel:9003 -alias=ariel evstrust.jks
```

5. Update the `config.xml` file of `server1`, adding trust keystore information to the `<ssl>` element and adding a `<use-secure-connections>` element, as shown in bold in the following snippet:

```
<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>
   <trust-store>./ssl/evstrust.jks</trust-store>
   <trust-store-pass>
      <password>secret</password>
   </trust-store-pass>
   <trust-store-type>JKS</trust-store-type>
   <ssl-protocol>TLS</ssl-protocol>
   <enforce-fips>false</enforce-fips>
   <need-client-auth>false</need-client-auth>
</ssl>

<use-secure-connections>
   <value>true</value>
</use-secure-connections>
```

The config file is located in the `config` subdirectory of the main server directory, such as `/oracle_cep/user_projects/domains/mydomain/server1/config/`.

6. Encrypt the cleartext password in the `<password>` of the `config.xml` file by using the encryption utility. See "The encryptMSAConfig Command Line Utility" on page 7-27.

Disabling Security

To disable security in a domain, add the `-disablesecurity` flag to the `java` command that starts Oracle CEP in the `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX) start script. This script is located in the server directory of your domain, such as *`ORACLE_CEP_HOME`*`/user_projects/domains/wlevs30_domain/defaultserver`.

The following snippet from the Windows `startwlevs.cmd` script shows in bold how to disable security:

```
%JAVA_HOME%\bin\java %DGC% %DEBUG% -Dwlevs.home=%USER_INSTALL_DIR%
-Dbea.home=%BEA_HOME% -jar "%USER_INSTALL_DIR%\bin\wlevs.jar" -disablesecurity
%1 %2 %3 %4 %5 %6
```

Locking Down the Server

This section describes how to lock down the server so that only HTTPS connections are allowed.

1. Ensure that SSL is configured for the server. See "Using SSL to Secure Network Traffic" on page 7-20 for details.
2. Remove the HTTP port configuration from the server's *`DOMAIN_DIR/servername`*`/config/config.xml file`, leaving only the configuration for the HTTPS port.

   For example, the following `config.xml` snippet shows a standard configuration in which both an HTTP and HTTPS port have been configured. The HTTP port is 9002 and the HTTPS port is 9003. The Jetty Server can be accessed using both ports. Only relevant parts of the `config.xml` file are shown:

   ```
   <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>
```

A modified `config.xml` file with HTTP access removed would be as follows:

```
<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 multi-server domain, be sure that SSL has been configured between the member servers. See "Configuring SSL In a Multi-Server Domain for Use By Visualizer" on page 7-21 for details.

Configuring Java SE Security

The Java SE platform defines a standards-based and interoperable security architecture that is dynamic and extensible. Security features — cryptography, authentication and authorization, public key infrastructure, and more — are built in. For further details, see Java SE Security Overview.

Oracle CEP supports Java SE security by using the following security policies:

- policy.xml—Defines the security policies of all the bundles that make up Oracle CEP. The first bundle set defines the policies for server-related bund;es; 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 Visualizer Web application.

Samples of the preceding files are shipped with the product and can be found in *ORACLE_CEP_HOME*/ocep_10.3/utils/security, where *ORACLE_CEP_HOME* refers to the directory in which you installed Oracle CEP, such as /oracle_home.

To enable all Java SE security features with Oracle CEP, follow these steps:

1. Stop the Oracle CEP server, if it is currently running.
2. Create the following two files in the *DOMAIN_DIR*/*servername*/config directory of your Oracle CEP server, where *DOMAIN_DIR* refers to the main Oracle CEP installation directory and *servername* refers to the name of your server, such as /oracle_cep/user_projects/domains/mydomain/myserver/config:
   - policy.xml
   - security.policy

   Copy the sample content from the preceding links into the files.
3. Edit the two security policy files to suit your needs.
4. Update the server startup script for your platform, startwlevs.cmd (Windows) or startwlevs.sh (UNIX), by adding the following three properties to the java command that actually starts the server:

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

   The server startup files are located in the *DOMAIN_DIR*/*servername* directory. For example:

   ```
   "%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. Update the *DOMAIN_DIR*/*servername*/config/config.xml file of your Oracle CEP server and edit the Jetty configuration 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 CEP server for the changes to take effect.

Security Command Line Utility Reference

Oracle CEP provides the a variety of command-line utilities for configuring security. See the following sections:

- "The cssconfig Command Line Utility" on page 7-26
- "The encryptMSAConfig Command Line Utility" on page 7-27
- "The passgen Command Line Utility" on page 7-28 (Deprecated)
- "The secgen Command Line Utility" on page 7-31 (Deprecated)

The cssconfig Command Line Utility

Use the `cssconfig` command line utility to generate a security configuration file (`security.xml`) that uses a password policy.

The `cssconfig` utility is located in the *`ORACLE_CEP_HOME`*`/ocep_10.3/bin` directory, where *`ORACLE_CEP_HOME`* is the main Oracle CEP installation directory, such as `d:\oracle_cep`. The utility comes in two flavors:

- `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:

```
prompt> $PATH_TO_KSH_BIN/ksh -c cssconfig.sh
```

where *PATH_TO_KSH_BIN* is the fully qualified path to the ksh program.

cssconfig Syntax

```
cssconfig -p propertyfile [-c configfile] -i inputkeyfile [-d]
```

where:

- *propertyfile* is a file that contains security configuratin properties. This option is required. property file provided by user to define what configuration s/he wants. See "Sample LDAP/DBMS Properties File" on page 7-7 for an example.
- *configfile* is the name of the generated file. This property is optional; default value is security.xml.
- *inputkeyfile* is the fully qualified name of the input key file used to generate the security configuration file. Set this optoin to the security-key.dat file in the config directory.
- -d enables debugging.

The encryptMSAConfig Command Line Utility

Use the encryptMSAConfig encryption command line utility to encrypt cleartext passwords, 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

The encryptMSAConfig utility is located in the *ORACLE_CEP_HOME*/ocep_10.3/bin directory, where *ORACLE_CEP_HOME* is the main Oracle CEP installation directory, such as d:\oracle_cep. The utility comes in two flavors:

- encryptMSAConfig.cmd (Windows)
- encryptMSAConfig.sh (UNIX)

encryptMSAConfig Syntax

```
encryptMSAConfig directory XML_file msainternal.dat_file
```

where:

- *directory* refers to the directory that contains the XML file which in turn contains a cleartext <password> element.
- *XML_file* refers to the name of your XML file.
- *msainternal.dat_file* parameter refers to the location of the .msainternal.dat file associated with your domain; this file is located in the *DOMAIN_DIR*/*servername* directory, where *DOMAIN_DIR* refers to the domain directory such as /oracle_cep/user_projects/domains/mydomain and *servername* refers to the server instance.

After you run the command, the value of the <password> element will be encrypted.

The passgen Command Line Utility

WARNING: The passgen command line utility has been deprecated as of Version 10.3 of Oracle CEP. This is because the Configuration Wizard automatically performs the required task for you.

Use the passgen command line utility to hash user passwords for addition to a security database.

The passgen utility is located in the *ORACLE_CEP_HOME*/ocep_10.3/bin directory, where *ORACLE_CEP_HOME* is the main Oracle CEP installation directory, such as d:\oracle_cep. The utility comes in two flavors:

- passgen.cmd (Windows)
- passgen.sh (UNIX)

passgen Syntax

```
passgen [-a algorithm] [-s saltsize] [-h] [-?] [password]*
```

where:

Option	Description	Default Value
`-a`	*`algorithm`* specifies the hash algorithm to use: • SHA-1 • MD2 • MD5 • SSHA • SHA-256 **Note:** The actual list of algorithms that can be set depends on the security providers plugged into the JDK.	If not specified, the default is SHA-1.
`-s`	*`saltsize`* is the number of salt characters added to ensure a unique hash string.	If not specified, the default is 4.
`-h, -?`	Displays command line options and exits.	
`password`	If passwords are specified on the command line they shall be hashed and printed out one per line in order from left to right. If no passwords are specified on the command line, then the tool shall prompt for passwords to hash interactively.	

Note: Windows operating systems must use the `.cmd` version of this utility, Unix platforms should use the `.sh` version.

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 passgen.sh
```

where *PATH_TO_KSH_BIN* is the fully qualified path to the `ksh` program.

Examples of Using passgen

The following sections provide examples that use the `passgen` utility:

- "Using passgen interactively" on page 7-30
- "Providing a Password on the Command Line" on page 7-30

Using passgen interactively

The following is an example of using the `passgen` utility interactively:

```
$ passgen
Password ("quit" to end): maltese
{SHA-1}LOtYvfQZj++4rV50AKpAvwMlQjqVd7ge
Password ("quit" to end): falcon
{SHA-1}u7NPQfgkHISr0tZUsmPrPmr3U1LKcAdP
Password ("quit" to end): quit
{SHA-1}2pPo4ViKsoNct3lTDoLeg9gHYZwQ47sV
```

In this mode, a password is entered and the resulting hashed version of the password is displayed. The hashed version of the password can then be entered into the password field of a security database.

Note: In example, the passwords are shown to be echoed to the screen for demonstration purposes. In most situations, the password would not be displayed unless your platform does not support invisible passwords.

Providing a Password on the Command Line

The following is an example using the `passgen` utility when providing the passwords to be hashed on the command line:

```
$ passgen maltese falcon
{SHA-1}g0PNXmJW0OBtp/GkHrhNAhpbjM+capNe
{SHA-1}2ivZnjnKD9fordC1YFkrVGf0DHL6SVP1
```

When multiple passwords are provided, they are hashed from left to right:

- `{SHA-1}g0PNXmJW0OBtp/GkHrhNAhpbjM+capNe` is hashed from maltese
- `{SHA-1}2ivZnjnKD9fordC1YFkrVGf0DHL6SVP1` is hashed from falcon.

The secgen Command Line Utility

WARNING: The `secgen` command line utility has been deprecated as of Version 10.3 of Oracle CEP. This is because the Configuration Wizard automatically performs the required task for you.

Use the `secgen` command line utility generates a security key or a security configuration file that uses encrypted passwords.

Note: This utility creates a security file that does not use a password policy; if you require a password policy, use the cssconfig command-line utility instead. See "The cssconfig Command Line Utility" on page 7-26.

The `secgen` utility is located in the *`ORACLE_CEP_HOME`*`/ocep_10.3/bin` directory, where *`ORACLE_CEP_HOME`* is the main Oracle CEP installation directory, such as `d:\oracle_cep`. The utility comes in two flavors:

- `secgen.cmd` (Windows)
- `secgen.sh` (UNIX)

Generating a File-Based Provider Configuration File

Use the following command line options to generate a file-based security provider configuration file.

```
secgen -F [-o outputfile] [-i inputkeyfile] [-e] [-P PropertyFilePath]
```

where:

Option	Description	Comments
`-F`	Generate a file-based security provider file; mutually exclusive with the `-k` option.	If not present, `-k` is assumed.
`-o`	*`outputfile`* is the name for the generated file.	Default output file name is `security.xml`.

Option	Description	Comments
`-i`	*`inputkeyfile`* is the fully qualified name of the input key file.	If not present, a default input key file named `security-key.dat` is expected.
`-e`	Enables unanimous adjudication during authorization.	
`-P`	*`PropertyFilePath`* is the fully qualified path to a `secgen` property file which you can use to customize provider configurations. See "Using the secgen Properties File" on page 7-32 for details.	A `SecGenTemplate.properties` template file is located at *`ORACLE_CEP_HOME`*`/ocep_10.3/bin` where *`ORACLE_CEP_HOME`* is the main installation directory of Oracle CEP, such as `/oracle_cep`.

Generating a Key File

Use the following command line options to generate a security key file.

```
secgen [-k] [-o outputfile]
```

where:

Option	Description	Comments
`-k`	Generate a key file; mutually exclusive with the `-F` option.	If not present, `-k` is assumed.
`-o`	*`outputfile`* is the name for the generated file.	Default output file name is `security-key.dat`.

Using the secgen Properties File

When running secgen, you can use the `-P` option to specify a property file to customize provider configurations. A `SecGenTemplate.properties` template file is located in *`ORACLE_CEP_HOME`*`/ocep_10.3/bin` where *`ORACLE_CEP_HOME`* is the main installation directory of Oracle CEP, such as `/oracle_cep`.

You specify cleartext passwords the property file; however, these passwords will be stored encrypted in the generated configuration file.

The following example shows a property file used for file based provider customization:

```
#File based provider related
file.atn.file.store.path=myfileatnstore.txt
file.atn.file.store.password=firewall
file.atn.user.password.style=HASHED
file.atn.file.store.encrypted=true
file.atz.file.store.path=filatz
file.atz.file.store.password=firewall
file.rm.file.store.path=filerm
file.rm.file.store.password=firewall
file.cm.file.store.path=filecm
file.cm.file.store.password=firewall
```

The legal values for `file.atn.user.password.style` are:

- `HASHED`
- `REVERSIBLEENCRYPTED`

Examples of Using secgen

The following example shows how to use the `secgen` utility to generate a key file with the name `myKeyFile.dat`:

```
prompt> secgen -k -o myKeyFile.dat
```

The following example shows how to use the `secgen` utility to generate a file-based security provider configuration file named `myConfigFile.xml` which also uses the previously generated key file, `myKeyFile.dat`, and a properties file named `mySecGen.properties`:

```
prompt> secgen -F -i myKeyFile.dat -o myConfigFile.xml -P
c:\msa\myMSAConfig\mySecGen.properties
```

Limitations of secgen

Windows operating systems must use the `.cmd` version of this utility, Unix platforms should use the `.sh` version.

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:

```
prompt> $PATH_TO_KSH_BIN/ksh -c secgen.sh
```

where *PATH_TO_KSH_BIN* is the fully qualified path to the `ksh` program.

CHAPTER 8

Configuring Jetty for Oracle Complex Event Processing

This section contains information on the following subjects:

Overview of Jetty Support in Oracle Complex Event Processing

Oracle Complex Event Processing (or *Oracle CEP* for short) supports Jetty as Java Web server to deploy HTTP servlets and static resources.

Oracle CEP support for Jetty is based on Version 1.2 the OSGi HTTP Service. This API provides ability to dynamically register and unregister `javax.servlet.Servlet` objects with the run time and static resources. This specification requires at minimum version 2.1 of the Java Servlet API.

Oracle CEP supports the following features for Jetty:

For details about configuring Jetty, see "Configuring a Jetty Server Instance" on page 8-4.

Servlets

In addition to supporting typical (synchronous) Java servlets, Oracle CEP supports asynchronous 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.

Network I/O Integration

Oracle CEP uses network I/O (NetIO) to configure the port and listen address of Jetty services.

Note: 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 CEP Jetty services use the Oracle CEP Work Manager to provide for scalable thread pooling. See "</config>" on page 8-9.

Note: Jetty provides its own thread pooling capability. However, Oracle recommends using the Oracle CEP self-tuning thread pool to minimize footprint and configuration complexity.

Work Managers

Oracle CEP allows you to configure how your application prioritizes the execution of its work. Based on rules you define and by monitoring actual run time performance, you can optimize the performance of your application and maintain service level agreements. You define the rules and constraints for your application by defining a work manager.

Understanding How Oracle CEP Uses Thread Pools

Oracle CEP uses is a single thread pool, in which all types of work are executed. Oracle CEP prioritizes work based on rules you define, and run-time metrics, including the actual time it takes to execute a request and the rate at which requests are entering and leaving the pool.

The common thread pool changes its size automatically 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 CEP increases the thread count. Similarly, if statistics indicate that fewer threads did not reduce throughput, Oracle CEP decreases the thread count.

Understanding Work Manager

Oracle CEP prioritizes work and allocates threads based on an execution model that takes into account 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 CEP uses these guidelines to assign pending work and enqueued requests to execution threads.

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

- `fairshare`—Specifies the average thread-use time required to process requests.

  For example, assume that Oracle CEP is running two modules. 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 CEP allocates 80% and 20% of the thread-usage time to `ModuleA` and `ModuleB`, respectively.

Note: The value of a fair share request class is specified as a relative value, not a percentage. Therefore, in the above example, if the request classes were defined as 400 and 100, they would still have the same relative values.

- `max-threads-constraint`—This 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 CEP does not schedule requests of this type until the number of concurrent executions falls below the limit. The Oracle CEP then schedules work based on the fair share or response time goal.

- `min-threads-constraint`—This 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 necessarily increase a fair share. This type of constraint has an effect primarily when the Oracle CEP instance is close to a deadlock condition. In that case, it the constraint causes Oracle CEP to schedule a request even if requests in the service class have gotten more than their fair share recently.

Configuring a Jetty Server Instance

You use the following configuration objects to configure an instance of the Jetty HTTP server in the `config.xml` file that describes your Oracle CEP domain:

- `<jetty>`: See "jetty Configuration Object" on page 8-4 for details.
- `<netio>`: See "netio Configuration Object" on page 8-5 for details.
- `<work-manager>`: See "work-manager Configuration Object" on page 8-6 for details.

Use the `<jetty-web-app>` configuration object to define a Web application in the Jetty instance; see "jetty-web-app Configuration Object" on page 8-6 for details.

See "Example Jetty Configuration" on page 8-8 for a sample of using each of the preceding configuration objects.

jetty Configuration Object

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

Table 8-1 Configuration Parameters for <jetty>

Parameter	Type	Description
`network-io-name`	`String`	The name of the NetIO service used. The NetIO service defines the port the server listens on. See "netio Configuration Object" on page 8-5 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 "work-manager Configuration Object" on page 8-6.
`scratch-directory`	`String`	The name of a directory where temporary files required for web apps, 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 Configuration Object

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

Table 8-2 Configuration Parameters for <netio>

Parameter	Type	Description
`name`	`String`	The name of this configuration object.

Table 8-2 Configuration Parameters for <netio>

Parameter	Type	Description
`port`	`int`	The listening port number.
`listen-address`	`String`	The address on which an instance of netio service listens for incoming connections. • It may 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. **Note:** The value of this parameter cannot be validated until the service has started.

work-manager Configuration Object

Use the parameters described in the following table to define a `<work-manager>` configuration object in your `config.xml` file.

Table 8-3 Configuration Parameters for <work-manager>

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 Configuration Object

Use the following configuration object to define a Web application for use by Jetty:

Table 8-4 Configuration Parameters for <jetty-web-app>

Parameter	Type	Description
`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 "Configuring a Jetty Server Instance" on page 8-4. If not specified, a directory is created at????.
`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 "Configuring a Jetty Server Instance" on page 8-4.
`name`	`String`	The name of this configuration object.

Developing Servlets for Jetty

Oracle CEP supports development of servlets for deployment to Jetty by creating a standard J2EE Web Application and configuring it using the "jetty-web-app Configuration Object" on page 8-6.

Web App Deployment

Oracle CEP supports 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 apps 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.

Example Jetty Configuration

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

Listing 8-1 Example Jetty Configuration

```
<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>
```

CHAPTER 9

Configuring JMX for Oracle Complex Event Processing

This section contains information on the following subjects:

Overview of JMX Support in Oracle Complex Event Processing

Oracle Complex Event Processing (or *Oracle CEP* for short) provides standards-based interfaces that are fully compliant with the Java Management Extensions (JMX) specification. Software vendors can use these interfaces to monitor Oracle CEP MBeans, to change the configuration of an Oracle CEP domain, and to and monitor the distribution (activation) of those changes to all server instances in the domain.

The `wlevs.Admin` utility uses JMX to connect to a server so you can manipulate its MBean instances, in particular to view, add, and update the EPL rules associated with the processors of a particular Oracle CEP application.

However, to use the `wlevs.Admin` utility, and the JMX interfaces in general, you must configure Oracle CEP with the JMX configuration information in the `config.xml` file.

Configuring JMX

You use the following configuration objects to configure an instance of the Jetty HTTP server in the `config.xml` file that describes your Oracle CEP domain:

- `<jmx>`: See "jmx Configuration Object" on page 9-2 for details.
- `<rmi>`: See "rmi Configuration Object" on page 9-2 for details.
- `<jndi-context>`: See "jndi-context Configuration Object" on page 9-3 for details.
- `<exported-jndi-context>`: See "exported-jndi-context Configuration Object" on page 9-4 for details

See "Example of Configuring JMX" on page 9-5 for a sample of using each of the preceding configuration objects.

jmx Configuration Object

The following table describes the configuration information for the `<jmx>` element in the `config.xml` file.

Table 9-1 Configuration Parameters for <jmx>

Parameter	Type	Description
`rmi-service-name`	`String`	The name of the RMI service with which the jmx server will register to receive calls.
`rmi-jrmp-port`	`int`	The port on which to listen for RMI JRMP JMX requests
`jndi-service-name`	`String`	The name of the JNDI service to which the jmx server will bind its object.
`rmi-registry-port`	`int`	The port on which to start the RMIRegistry

rmi Configuration Object

The Oracle CEP RMI service provides:

- 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.

The following table shows the parameters of the `<rmi>` configuration object that you use to export server-side objects to remote clients.

Table 9-2 Configuration Parameters for <rmi>

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 "Overview of Jetty Support in Oracle Complex Event Processing" on page 8-1).
`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 Configuration Object

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 CEP, that initializes the JNDI environment to support the factory manager bundle.

Use the following configuration object to configure the `<jndi-context>` configuration object.

Table 9-3 Configuration Parameters for <jndi-context>

Parameter	Type	Description
`default-provider`	`boolean`	If `true`, the default Oracle CEP JNDI provider is used. Default value is `true`.
`name`	`String`	The name of this configuration object.

exported-jndi-context Configuration Object

Note: Requires a configured "jndi-context Configuration Object" on page 9-3.

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 9-4 Configuration Parameters for <exported-jndi-context>

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 Configuration Object" on page 9-2.
`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.

Example of Configuring JMX

The following `config.xml` snippet shows an example of configuring JMX; only relevant parts of the file are shown.

Listing 9-1

```
<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>
    <rmi-registry-port>10099</rmi-registry-port>
    <rmi-jrmp-port>9999</rmi-jrmp-port>
  </jmx>

</config>
```

CHAPTER 10

Configuring Access to a Relational Database

This section contains information on the following subjects:

Note: This section describes in general how to configure database access using JDBC data sources. Oracle CEP includes a Type 4 JDBC drivers from DataDirect for SQL Server; for specific information about them, see Oracle CEP Type 4 JDBC Drivers.

Overview of Database Access from an Oracle CEP Application

Oracle Complex Event Processing, or *Oracle CEP* for short, supports Java Database Connectivity (JDBC) 3.0 for relational database access.

The JDBC API provides a standard, vendor-neutral mechanism for connecting to and interacting with database servers and other types of tabular resources that support the API. The JDBC `javax.sql.DataSource` interface specifies a database connection factory that is implemented by a driver. Instances of `DataSource` objects are used by applications to obtain database connections (instances of `java.sql.Connection`). After obtaining a connection, an application interacts with the resource by sending SQL commands and receiving results.

WebLogic Event Server provides the following JDBC drivers:

- Oracle 10.2.0 thin driver (packaged in the *ORACLE_CEP_HOME*/modules/com.bea.oracle.ojdbc5_11.1.0.6.jar and *ORACLE_CEP_HOME*/modules/com.bea.oracle.ojdbc6_11.1.0.6.jar files)
- SQL Server Type 4 JDBC Driver from DataDirect (see "Type 4 JDBC Driver for SQL Server from DataDirect" on page 10-2)

WebLogic Event Server also provides a `DataSource` abstraction that encapsulates a JDBC driver `DataSource` object and manages a pool of pre-established connections.

Type 4 JDBC Driver for SQL Server from DataDirect

Oracle CEP provides a Type 4 JDBC driver from DataDirect for high-performance JDBC access to the SQL Server database. The Type 4 JDBC driver is optimized for the Java environment, allowing you to incorporate Java technology and extend the functionality and performance of your existing system. For detailed information about using the two drivers, see Oracle CEP Type 4 JDBC Drivers.

The Oracle CEP Type 4 JDBC drivers from DataDirect are proven drivers that:

- Support performance-oriented and enterprise functionality such as distributed transactions, savepoints, multiple open result sets and parameter metadata.
- Are Java EE Compatibility Test Suite (CTS) certified and tested with the largest JDBC test suite in the industry.
- Include tools for testing and debugging JDBC applications.

Supported Databases

Table 10-1 shows the databases supported by each of the Oracle CEP Type 4 JDBC drivers.

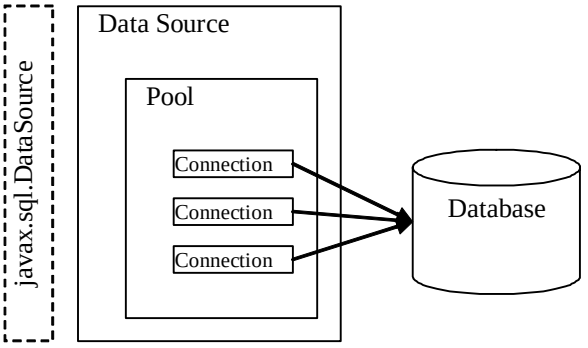
Table 10-1 Supported Databases

Driver	Supported Databases
SQL Server	• Microsoft SQL Server 2005 • Microsoft SQL Server 2000 • Microsoft SQL Server 2000 Desktop Engine (MSDE 2000) • SQL Server 2000 Enterprise Edition (64-bit) • Microsoft SQL Server 7.0

Description of Oracle CEP Data Sources

Oracle CEP `DataSource` provides a JDBC data source connection pooling implementation that supports the Java Database Connectivity (JDBC 3.0) specification. Applications reserve and release `Connection` objects from a data source using the standard `DataSource.getConnection` and `Connection.close` APIs respectively.

Figure 10-1 Data Source



You are required to configure an Oracle CEP `DataSource` in the server's `config.xml` file if you want to access a relational database from an EPL `rule`; for details, see Configuring the Complex Event Processor. 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.

Data Source Configuration

The Oracle CEP `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 CEP domain using the Configuration Wizard, you can optionally configure a JDBC data source that uses one of the two DataDirect JDBC drivers; in this case the wizard updates the `config.xml` file for you. You configure the data source with basic information, such as the database you want to connect to and the connection username and password. You can also use the Configuration Wizard to update an existing server in a domain and add new data sources. For details about using the Configuration Wizard, see Creating and Updating an Oracle CEP Standalone-Server Domain.

You can also update the config.xml file manually by adding a <data-source> element. The following snippet shows a sample data source configuration:

```
<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>
```

A data source depends on the availability of a local transaction manager, which you configure using the `<transaction-manager>` element of `config.xml` as shown above. The transaction manager in turn depends on a configured RMI object, as described in "rmi Configuration Object" on page 9-2.

For the full list of child elements of the `<data-source>` element, in particular the `<connection-pool-params>` and `<data-source-params>` elements, see the Schema.

Configuring Access to a Database using the Type 4 JDBC Drivers from Data Direct

The two type 4 JDBC drivers from DataDirect (Oracle and SQL Server) are automatically installed with Oracle CEP and ready to use.

1. Configure the data source in the server's `config.xml` file by either using the Configuration Wizard or updating the `config.xml` file manually.

   For details, see "Data Source Configuration" on page 10-4.

2. If Oracle CEP is running, restart it so it reads the new data source information. See Stopping and Starting the Server.

Configuring Access to a Database Using Your Own Database Drivers

Follow these steps to configure and use your own JDBC driver with Oracle CEP:

1. Update the server start script in the server directory of your domain directory so that Oracle CEP finds the appropriate JDBC driver JAR file when it boots up.

   The name of the server start script is `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX), and the script is located in the server directory of your domain directory. The out-of-the-box sample domains are located in *`ORACLE_CEP_HOME`*`/ocep_10.3/samples/domains`, and the user domains are located in *`ORACLE_CEP_HOME`*`/user_projects/domains`, where *`ORACLE_CEP_HOME`* refers to the Oracle CEP installation directory, such as `d:\oracle_cep`.

Update the start script by adding the `-Xbootclasspath/a` option to the Java command that executes the `wlevs_3.0.jar` file. Set the `-Xbootclasspath/a` option to the full pathname of the JDBC driver you are going to use.

For example, if you want to use the Windows Oracle thin driver, update the `java` command in the start script as follows (updated section shown in bold):

```
  %JAVA_HOME%\bin\java -Dwlevs.home=%USER_INSTALL_DIR%
-Dbea.home=%BEA_HOME%
-Xbootclasspath/a:%USER_INSTALL_DIR%\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
```

In the example, `%USER_INSTALL_DIR%` points to *`ORACLE_CEP_HOME`*`\ocep_10.3`.

2. Configure the data source in the server's `config.xml` file by either using the Configuration Wizard or updating the `config.xml` file manually.

   For details, see "Data Source Configuration" on page 10-4.

3. If Oracle CEP is running, restart it so it reads the new `java` option and data source information. See Stopping and Starting the Server.

CHAPTER 11

Configuring the HTTP Publish-Subscribe Server

This section contains information on the following subjects:

Overview of HTTP Publish-Subscribe Servers

An *HTTP Publish-Subscribe Server* (for simplicity, also called *pub-sub server* in this document) is a mechanism whereby Web clients subscribe to channels and then publish messages to these channels using asynchronous messages over HTTP.

The simple request/response nature of a standard Web application requires that all communication be initiated by the client; this means that the server can only push updated data to its clients if it receives an explicit request. This mechanism is adequate for traditional applications in which data from the server is required only when a client requests it, but inadequate for dynamic real-time applications in which the server must send data even if a client has not explicitly requested it. The client can use the traditional HTTP pull approach to check and retrieve the latest data at regular intervals, but this approach is lacking in scalability and leads to

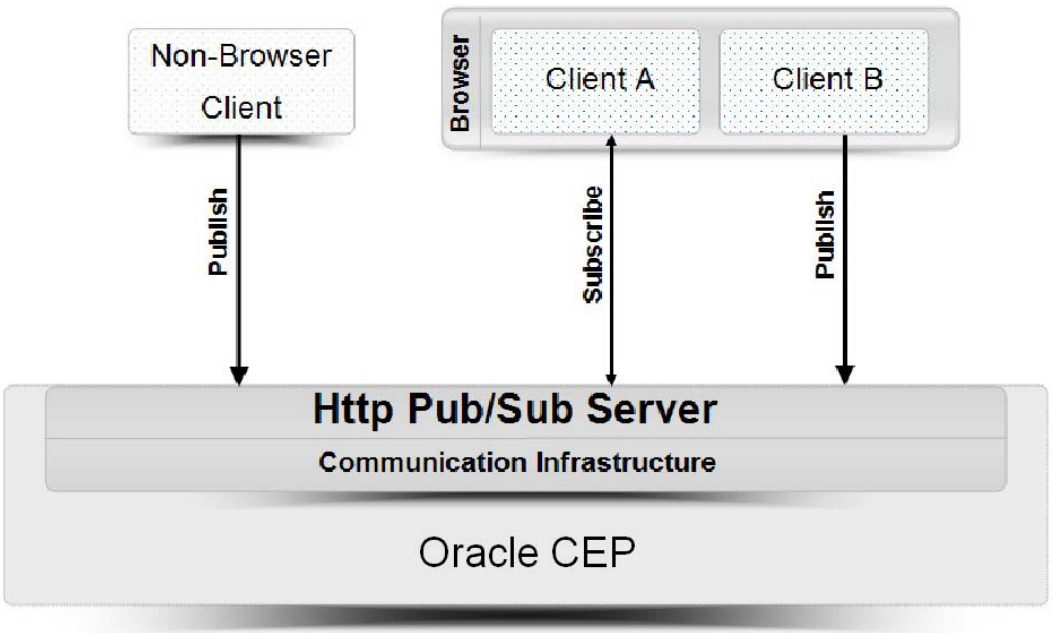
high network traffic because of redundant checks. The HTTP Publish-Subscribe Server solves this problem by allowing clients to subscribe to a channel (similar to a topic in JMS) and receive messages as they become available.

The 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, which are named destinations or sources of events. Registered clients, or the pub-sub server itself, then publishes messages to these channels which in turn any subscribed clients receive.

The pub-sub server can communicate with any client that can understand the Bayeux protocol. The pub-sub server is responsible for identifying clients, negotiating trust, exchanging Bayeux messages, and, most importantly, pushing event messages to subscribed clients.

The following figure describes the basic architecture of the pub-sub server included in Oracle CEP.

Figure 11-1 HTTP Publish-Subscribe Server in Oracle CEP



How the HTTP Pub-Sub Server Works

There is a one-to-one relationship between a servlet and a pub-sub server; in other words, each servlet has access to one unique pub-sub server. Each pub-sub server has its own list of channels. The servlet uses a context object to get a handle to its associated pub-sub server.

In Oracle CEP, 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 pub-sub server, specify the transport and other parameters. You then use Visualizer to add new channels and configure security for the channels.

Oracle CEP application developers can optionally use the built-in HTTP pub-sub adapters to publish and subscribe to channels within their applications. If, however, developers need the pub-sub server to perform additional steps, such as monitoring, collecting, or interpreting incoming messages from clients, then they must use the server-side pub-sub server APIs to program this functionality.

For Web 2.0 Ajax clients (e.g. Dojo) or RIA (e.g. Adobe Flex) to communicate with the 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.

HTTP Pub-Sub Server Support in Oracle CEP

Every Oracle CEP server includes a default HTTP pub-sub server. This server is used internally by Visualizer and by the Record and Playback example. You can either use the default pub-sub server for your own Web 2.0 application or create a new one.

The default pub-sub server has the following properties:

- Pub-sub server URL— `http://`*`host`*`:`*`port`*`/pubsub`, where *`host`* and *`port`* refer to the computer on which Oracle CEP is running and the port number to which it listens, respectively.
- Transport: Uses long-polling transport.
- Allows clients to publish messages to a channel without having explicitly connected to the pub-sub server.
- Includes the following three channels used internally by Visualizer; do not delete these channels:
  - `/evsmonitor`

- `/evsalert/`
- `/evsdomainchange`

For details about configuring the default pub-sub server, or creating a new one, see "Configuring an Existing HTTP Publish-Subscribe Server" on page 11-4.

Oracle CEP also includes two built-in adapters that easily harness HTTP pub-sub server functionality in your applications. By adding these adapters to your application you can both publish messages or subscribe to a server to receive messages, using either the local HTTP pub-sub server or a remote one. For details, see Using and Creating HTTP Publish-Subscribe Adapters.

Configuring an Existing HTTP Publish-Subscribe Server

The following procedure describes how to configure an existing HTTP pub-sub server. See "Creating a New HTTP Publish-Subscribe Server" on page 11-6 for a full example from the `config.xml` of a configured HTTP pub-sub server.

1. If the Oracle CEP server is running, stop it. See Stopping and Starting the Server.
2. Using your favorite XML editor, open the Oracle CEP server's `config.xml` file.

   This file is located in the *`DOMAIN_DIR`*`/`*`servername`*`/config` directory, where *`DOMAIN_DIR`* refers to the domain directory and *`servername`* refers to the name of the server, such as `/oracle_cep/user_projects/myDomain/defaultserver/config`.
3. Search for the `<http-pubsub>` element that corresponds to the HTTP pub-sub server you want to configure. For example, the default pub-sub server is as follows:

```
<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 in turn is a child element of `<http-pubsub>`) with pub-sub server configuration as required. For the full list of possible elements, see the XSD Schema. The following are the most common configuration options:

– Add a `<supported-transport>` element to specify the transport; currently the two supported transports are long-polling and callback-polling. The format of this element is as follows:

```
<server-config>
    <supported-transport>
        <types>
            <element>long-polling</element>
        </types>
    </supported-transport>
...
</server-config>
```

– Add a `<publish-without-connect-allowed>` element to specify whether clients can publish messages without having explicitly connected to the pub-sub server; valid values are `true` or `false`:

```
<server-config>
...

<publish-without-connect-allowed>true</publish-without-connect-allow
ed>
</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. See "work-manager Configuration Object" on page 8-6 for details.

```
<server-config>
...
   <work-manager>myWorkManager</work-manager>
</server-config>
```

– Add a `<client-timeout-secs>` element to specify the number of seconds after which the 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 and restart Oracle CEP.
6. Use Visualizer to configure or add channels, and to configure security for the channels. See:
   – Configuring HTTP Publish-Subscribe Server Channels
   – Configuring Security for the HTTP Publish-Subscribe Channels

Creating a New HTTP Publish-Subscribe Server

The following procedure describes how to create a new HTTP pub-sub server. See "Creating a New HTTP Publish-Subscribe Server" on page 11-6 for a full example from the `config.xml` of a configured HTTP pub-sub server.

1. If the Oracle CEP server is running, stop it. See Stopping and Starting the Server.
2. Using your favorite XML editor, open the Oracle CEP server's `config.xml` file.

   This file is located in the *`DOMAIN_DIR`*`/`*`servername`*`/config` directory, where *`DOMAIN_DIR`* refers to the domain directory and *`servername`* refers to the name of the server, such as `/oracle_cep/user_projects/myDomain/defaultserver/config`.
3. Add a `<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>
```

   Set the `<name>` element to the internal name of the HTTP pub-sub server. Set the `<path>` element to the string that you want to appear in the URL for connecting to the HTTP pub-sub server. 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 pub-sub server configuration as required. For the full list of possible elements, see the XSD Schema. The following are the most common configuration options:

   – Add a <supported-transport> element to specify the transport; currently the two supported transports are long-polling and callback-polling. The format of this element is as follows:

   ```
   <server-config>
       <supported-transport>
           <types>
               <element>long-polling</element>
           </types>
       </supported-transport>
   ...
   </server-config>
   ```

   – Add a <publish-without-connect-allowed> element to specify whether clients can publish messages without having explicitly connected to the pub-sub server; valid values are true or false:

   ```
   <server-config>
   ...

   <publish-without-connect-allowed>true</publish-without-connect-allow
   ed>
   </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. See "work-manager Configuration Object" on page 8-6 for details.

   ```
   <server-config>
   ...
       <work-manager>myWorkManager</work-manager>
   </server-config>
   ```

   – Add a <client-timeout-secs> element to specify the number of seconds after which the 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>
```

6. Update the `<channels>` child element with at least one channel pattern. Channel patterns always begin with a forward slash. These channels are what clients subscribe to either publish or receives messages. Add a channel pattern as shown:

```
<channels>
   <element>
     <channel-pattern>/mychannel</channel-pattern>
   </element>
</channels>
```

7. Save the `config.xml` file and restart Oracle CEP.
8. Use Visualizer to add more channels, and to configure security for the channels. See:
   - Configuring HTTP Publish-Subscribe Server Channels
   - Configuring Security for the HTTP Publish-Subscribe Channels

Example of Configuring the HTTP Publish-Subscribe Server

The following snippet of the `config.xml` file shows the configuration of the default HTTP pub-sub server present in every Oracle CEP server:

```
<?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>pubsub</name>
      <path>/pubsub</path>
      <pub-sub-bean>
        <server-config>
          <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>
...
</ns2:config>
```

CHAPTER 12

Configuring Logging and Debugging

This section contains information on the following subjects:

Configuration Scenarios

System administrators and developers configure logging output and filter log messages to troubleshoot errors or to receive notification for specific events.

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` severity level or higher.

- Specify a default logging level for the entire Oracle CEP server, and then have a specific Oracle CEP module override the default logging level. For example, the default logging level of the server could be WARNING while the logging level of the CEP module is DEBUG.
- Configure a logging level for a user-application deployed to Oracle CEP. In this case, the application must use the Commons Apache Logging Framework if the application is required to output log messages into the single server-wide log file to which the modules of the server itself also log their messages.

Overview of Logging Services Configuration

This release 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`.

- "Setting the Log Factory" on page 12-2
- "Using Log Severity Levels" on page 12-3
- "Log Message Format" on page 12-5
- "OSGI Framework Logger" on page 12-5

See `http://jakarta.apache.org/commons/logging/apidocs/index.html`.

Setting the Log Factory

The following provides information on 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.commo
ns.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.LogFactoryImpl
  ```

  or the equivalent programming would be:

  ```
  System.setProperty(LogFactory.FACTORY_PROPERTY, "org.apache.commons.log
  ging.impl.LogFactoryImpl");
  ```

Using 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 to highest 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 CEP 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 12-1 lists the severity levels of log messages.

Table 12-1 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.
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.

Log Message Format

The system writes a message to sdtout and the specified log file, consisting of the Timestamp, Severity, Subsystem, and the Message, along with the stacktrace if any. Each attribute is contained between angle brackets.

The following is an example of a message in the server log file:

```
<May 02, 2007 10:46:51 AM EST> <Notice> <CommonTestSubsystem> <BEA-123456>
<Another Commons test message>
```

Format of Output to Standard Out and Standard Error

When the system writes a message to standard out, the output does not include the #### prefix and does not include the Server Name, Machine Name, Thread ID, User ID, Transaction ID, Diagnostic Context ID, and Raw Time Value fields.

The following is an example of how the message from the previous section would be printed to standard out:

```
<Sept 22, 2004 10:51:10 AM EST> <Notice> <WebLogicServer> <BEA-000360>
<Server started in RUNNING mode>
```

In this example, the message attributes are: Locale-formatted Timestamp, Severity, Subsystem, Message ID, and Message Text.

OSGI Framework Logger

Oracle CEP 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.

How to Use the Commons Logging API

To use Commons Logging:

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.

Listing 12-1 Commons Code Example

```
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");
  }
}
```

Configuring the Oracle CEP Logging Service

The following sections provide information on configuring Oracle CEP logging:

logging-service

This section provides information on the `<logging-service>` configuration object:

Table 12-2 Configuration Parameters for <logging-service>

Parameter	Type	Description
`log-file-config`	`String`	The configuration of the log file and its rotation policies. See "log-file" on page 12-9.
`stdout-config`	`String`	The name of the `stdout` configuration object used to configure `stdout` output. See "log-stdout" on page 12-8.
`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`, `Trace`.

Table 12-2 Configuration Parameters for <logging-service>

Parameter	Type	Description
`name`	`String`	The name of this configuration object.
logger-severity-properties	One or more `<entry>` child elements.	List of name-value pairs, enclosed in an `<entry>` element, that list individual modules and their logging severity. These severities override the default severity of the Oracle CEP server. See "Configuring Severity for an Individual Module" on page 12-11.

log-stdout

This section provides information on the `log-stdout` configuration object:

Table 12-3 Configuration Parameters for log-stdout

Parameter	Type	Description
`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`.

Table 12-3 Configuration Parameters for log-stdout

Parameter	Type	Description
`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`.
`name`	`String`	The name of this configuration object.

log-file

This section provides information on the `log-file` configuration object:

Table 12-4 Configuration Parameters for log-file

Parameter	Type	Description
`number-of-files-limited`	`Boolean`	If `true`, old rotated files are deleted. Default is `false`.
`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`

Table 12-4 Configuration Parameters for log-file

Parameter	Type	Description
`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`.
`base-log-file-name`	`String`	The log file name. Default value is `server.log`.
`rotate-log-on-startup-enabled`	`Boolean`	If `true`, the log file is rotated on startup. Default value is `true`.
`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.
`name`	`String`	The name of this configuration object.

Configuring Severity for an Individual Module

Individual modules of Oracle CEP can specify their logging severity. This severity overrides the default logging severity of Oracle CEP, which is `Notice`. You do this by specifying the name of the module package and its logging severity as `<logger-severity-properties>` element within the log configuration in the server's `config.xml` file. Multiple packages or loggers can be specified in the `<logger-severity-properties>` element as name value pairs.

The following sample `config.xml` shows how to change the logging severity for a number of modules of Oracle CEP:

```
<logging-service>
  <name>myLogService</name>
  <logger-severity>Warning</logger-severity>
  <logger-severity-properties>
    <entry>
       <key>com.bea.wlevs.ede</key>
       <value>Debug</value>
    </entry>
    <entry>
       <key>sample.HelloWorld</key>
       <value>Debug</value>
    </entry>
    <entry>
       <key>com.bea.wlevs.cep.core.EPRuntimeImpl</key>
       <value>Debug</value>
    </entry>
  </logger-severity-properties>
  <stdout-config>myStdoutConfig</stdout-config>
  <log-file-config>myLogFileConfig</log-file-config>
</logging-service>
```

In the example, the default logging severity for the server is `WARNING`. However, the logging severity for any modules in the `com.bea.wlevs.ede` package, the `sample.HelloWorld` user application, and the `com.bea.wlevs.cep.core.EPRuntimeImpl` class is `DEBUG`.

Debug

The following sections provide information on how to use the Oracle CEP debugging feature:

Configuring debug using System Properties

You can set a system property on the Java command line by using the following steps:

1. Create a property by prepending `-D` to the flag
2. Turn the flag on by setting the property to `true`.

For example: `-Dcom.bea.core.debug.DebugSDS=true`

Configuring debug using a Configuration File

Use the following steps to configure debugging from a configuration file:

1. Create an XML tag by dropping "`com.bea.core.debug.`" from the flag name.
2. Turn the flag on by setting the flag to `true`.
3. Wrap with `debug-properties` tag.
4. Set `logger-severity` to `Debug` in the logging service stanza.
5. Set `stdout-severity` to `Debug` in the stdout configuration stanza.

See Example Debug Configuration.

Supported Debug Flags

The following table provides the supported debug flags for this release:

Table 12-5 Debug Flags

Debug Flag	Description
`com.bea.core.debug.DebugSDS`	Simple Declarative Services
`com.bea.core.debug.DebugSDS.stdout`	SDS debug strings go to stdout

Table 12-5 Debug Flags

Debug Flag	Description
com.bea.core.debug.DebugServiceHelper	Service Helper
com.bea.core.debug.DebugServiceHelper.stdout	Service Helper debug strings go to stdout
com.bea.core.debug.servicehelper.dumpstack	Dump stack traces when Service Helper times 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.DebugCM	Configuration Manager
com.bea.core.debug.DebugCM.stdout	Configuration Manager 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
com.bea.core.debug.DebugCSS	CSS
com.bea.core.debug.DebugCSS.stdout	CSS debug strings go to stdout
com.bea.core.debug.DebugBootBundle	Boot Debugging
com.bea.core.debug.DebugBootBundle.stdout	Boot Debugging debug strings go to stdout
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

Table 12-5 Debug Flags

Debug Flag	Description
com.bea.core.debug.DebugJTAHealth	JTA Health
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
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.DebugStoreAdmin	Store Administration
com.bea.core.debug.DebugStoreIOPhysical	Store IOPhysical
com.bea.core.debug.DebugStoreIOPhysicalVerbose	Store IOPhysicalVerbose
com.bea.core.debug.DebugStoreIOLogical	Store IOLogical

Table 12-5 Debug Flags

Debug Flag	Description
com.bea.core.debug.DebugStoreIOLogicalBoot	Store IOLogicalBoot
com.bea.core.debug.DebugStoreXA	Store XA
com.bea.core.debug.DebugStoreXAVerbose	Store XAVerbose
com.bea.core.debug.DebugConfigurationRuntime	Runtime information from the Runtime MBeans
com.bea.core.debug.DebugJDBCInternal	JDBC Internal
com.bea.core.debug.DebugJTAJDBC	JTA JDBC
com.bea.core.debug.DebugJDBCSQL	JDBC SQL
com.bea.core.debug.DebugJDBCRMI	JDBC RMI
com.bea.core.debug.DebugJDBCConn	JDBC Connection
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 stdout

Example Debug Configuration

The following code provides an example debug configuration to turn on Simple Declarative Services (SDS) debugging in the `config.xml file`:

Listing 12-2 Example debug Configuration

```
<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>
```

Log4j

The following sections provide information on using Log4j:

About Log4j

Log4j is an open source tool developed for putting log statements in your application. Log4j has three main components:

- "Loggers" on page 12-17
- "Appenders" on page 12-17
- "Layouts" on page 12-17

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/docs/`
- `http://logging.apache.org/log4j/docs/api/index.html`
- Short introduction to log4j at `http://logging.apache.org/log4j/docs/manual`.

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 namespace hierarchy and inherit behavior from their ancestors in the hierarchy.

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.

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.

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:

Listing 12-3 Example 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
```

Enabling 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.Log4JL
ogger
```

```
-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.