Oracle® Complex Event Processing

Administration and Configuration Guide

Release 3.0

July 2008

Alpha/Beta Draft

ORACLE

Oracle Complex Event Processing Administration and Configuration Guide, Release 3.0

Contents

1. Introduction and Roadmap

2. Creating and Updating an Oracle CEP Domain

6. Configuring Security for Oracle Complex Event Processing

7. Configuring Jetty for Oracle Complex Event Processing

8. Configuring JMX for Oracle Complex Event Processing

9. Configuring Access to a Relational Database

10. Configuring the HTTP Publish-Subscribe Server

11. 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*

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 Domain," describes how to create a new Oracle CEP domain.
- Chapter 3, "Configuring Oracle Complex Event Processing," describes how to start and stop an Oracle CEP server, as well as how to configure it.
- Chapter 4, "Configuring and Using Oracle CEP Clustered 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 6, "Configuring Security for Oracle Complex Event Processing," provides information about configuring various types of security for Oracle CEP.

- Chapter 7, “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 8, “Configuring JMX for Oracle Complex Event Processing,” describes how to configure some features of the microServices Architecture, in particular RMI, JNDI, and JMX.
- Chapter 9, “Configuring Access to a Relational Database,” describes how to configure access to a relational database.
- Chapter 10, “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 11, “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.

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 three 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 *`WLEVS_HOME`*`\samples\domains\helloworld_domain`, where *`WLEVS_HOME`* refers to the top-level Oracle CEP directory, such as `c:\beahome\wlevs30`.

The HelloWorld Java source code is located in *WLEVS_HOME*\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 *WLEVS_HOME*\samples\domains\fx_domain, where *WLEVS_HOME* refers to the top-level Oracle CEP directory, such as c:\beahome\wlevs30.

  The ForeignExchange Java source code is located in *WLEVS_HOME*\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 *WLEVS_HOME*\samples\domains\signalgeneration_domain, where *WLEVS_HOME* refers to the top-level Oracle CEP directory, such as c:\beahome\wlevs30.

  The Signal Generation Java source code is located in *WLEVS_HOME*\samples\source\applications\signalgeneration.

CHAPTER 2

Creating and Updating an Oracle CEP Domain

This section contains information on the following subjects:

Overview of Oracle Complex Event Processing Domains

A *domain* is the basic administration unit for Oracle Complex Event Processing (or *Oracle CEP* for short). It consists of one or more Oracle CEP server instances, zero or more deployed applications, and logically related resources and services that are managed, collectively, as one unit. If the domain contains more than one server instance, they can be clustered together. For details, see "Configuring and Using Oracle CEP Clustered Domains" on page 4-1.

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 *BEA_HOME*`/user_projects/domains` directory, where *BEA_HOME* refers to the parent directory of the main Oracle CEP installation directory such as `d:/beahome`. 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. Once you have created a domain with a single default server, you can use the Configuration Wizard to create additional servers in the domain.

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.
- `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 add a new server to an existing domain, or update an existing server to reconfigure its administration user, listen ports, and JDBC configuration.

Creating a Domain Using the Configuration Wizard

To create an Oracle CEP domain, use the Configuration Wizard. This tool guides you through the entire process, allowing you to customize the domain to more closely match your particular environment by:

- Adding a default server.
- Configuring the server's administration user and password.
- Configuring 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.
- Configuring the server's listen port, the JMX RMI registry listen port, or the JMX RMI JRMP listen port.

- Configuring 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 ment:

```
Start > All Programs > Oracle CEP 3.0 > 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 *`WLEVS_HOME`*`/common/bin` directory, where *`WLEVS_HOME`* refers to the main Oracle CEP installation directory, such as `/beahome_wlevs/wlevs30`:

   ```
   prompt> cd /beahome_wlevs/wlevs30/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 particualr:
   - 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 listen port for the JMX RMI registry, or the port on which to start the RMI registry. Default is 1099.
   - 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: *BEA_HOME*/user_projects/domains (UNIX) or *BEA_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:\bea_wlevs\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 *WLEVS_HOME*/common/bin directory, where *WLEVS_HOME* refers to the main Oracle CEP installation directory, such as /beahome_wlevs/wlevs30:

   ```
   prompt> cd /beahome_wlevs/wlevs30/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:\bea_wlevs\user_projects\domains` directory, update the corresponding `<data-value>` element as follows

```
<data-value name="DOMAIN_LOCATION"
            value="C:\bea_wlevs\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 or update 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 default server in this domain. This name will also be used as the name of the directory that contains the default 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 *`BEA_HOME`*`/user_projects/domains`, where *`BEA_HOME`* refers to the top-level installation directory, such as `c:/beahome`.
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.
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.

Table 2-1 Values for the silent.xml File

For this data-value name...	Enter the following value...
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:\bea_wlevs\user_projects\domains" />

    <data-value name="NETIO_PORT" value="9002" />
    <data-value name="RMI_REGISTRY_PORT" value="1099" />
```

```
    <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 Wixard 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.
```

Adding New Servers to an Existing Domain Using the Configuration Wizard

Use the Configuration Wizard to add new servers to an existing domain. The procedure is similar to creating a new domain, so be sure you read "Creating a 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:\bea_wlevs\user_projects\domains\mydomain`.
- The domain includes a single server called `defaultserver` and the server files are located in the `C:\bea_wlevs\user_projects\domains\mydomain\defaultserver` directory.
- You want to create a new server in the `mydomain` domain called `productionServer`.

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. *Be sure all this information is different from that of the other servers in the domain 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 `productionServer`.
   - 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.
7. If you chose to change the default JDBC data sourceconfiguration, 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:\bea_wlevs\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:\bea_wlevs\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.
- Set the SERVER_NAME option to the name of the *new* server you want to add to the existing domain.
- 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.

Based on the assumptions described in "Adding New Servers to an Existing Domain 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="createDomain" />
    <data-value name="USERNAME" value="wlevs" />
    <data-value name="PASSWORD" value="wlevs" />

    <data-value name="SERVER_NAME" value="productionServer" />
    <data-value name="DOMAIN_NAME" value="mydomain" />
    <data-value name="DOMAIN_LOCATION"
value="C:\bea_wlevs\user_projects\domains" />

    <data-value name="NETIO_PORT" value="9102" />
    <data-value name="RMI_REGISTRY_PORT" value="1199" />
    <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>
```

Updating an Existing Server Using the Configuration Wizard

Use the Configuration Wizard to update an existing servers in a domain. The procedure has similarities with creating a new domain and default server, so be sure you read "Creating a 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 ports
- 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:\bea_wlevs\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:\bea_wlevs\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:\bea_wlevs\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 2-13, 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:\bea_wlevs\user_projects\domains\mydomain\productionServer" />

  <data-value name="NETIO_PORT" value="9102" />
  <data-value name="RMI_REGISTRY_PORT" value="1199" />
  <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 *`WLEVS_HOME`*`/samples/domains/helloworld_domain/defaultserver`, where *`WLEVS_HOME`* refers to the main Oracle CEP installation directory, such as `/beahome_wlevs/wlevs30`.

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

   ```
   JAVA_HOME=BEA_HOME_WLEVS/jrockit-R27.6.0-23-1.5.0_15 (UNIX)
   set JAVA_HOME=BEA_HOME_WLEVS\jrockit-R27.6.0-23-1.5.0_15 (Windows)
   ```

   where *`BEA_HOME_WLEVS`* refers to the installation directory of Oracle CEP 3.0, such as `/beahome_wlevs` (UNIX) or `c:\beahome_wlevs` (Windows).

   If using the Oracle JRockit JDK installed with Oracle JRockit Real Time 2.0, the `JAVA_HOME` variable should be set as follows:

   ```
   JAVA_HOME=BEA_HOME_WLRT/jrockit-realtime20_150_11 (UNIX)
   set JAVA_HOME=BEA_HOME_WLRT\jrockit-realtime20_150_11 (Windows)
   ```

   where *`BEA_HOME_WLRT`* refers to the installation directory of Oracle JRockit Real Time 2.0, such as `/beahome_wlrt` (UNIX) or `c:\beahome_wlrt` (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:\bea_wlevs\wlevs30\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 2.0, 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:\bea_wlevs\wlevs30\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:1099/jmxrmi
   ```

   In the example, the host is `ariel` and the JMX port is `1099`.

   See Table 5-1, "Connection Arguments," on page 5-6 for additional details about the `-url` argument.

Next Steps

After creating your own Oracle CEP domain:

- Optionally configure the server and domain. See "Overview of Configuring Oracle Complex Event Processing" on page 3-1.
- 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, and how they all fit together.
- Deploy your new, or existing, Oracle CEP application to the domain. See Deploying Oracle CEP Applications.

CHAPTER 3

Configuring Oracle Complex Event Processing

This section contains information on the following subjects:

Overview of Configuring Oracle Complex Event Processing

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 2-15 for details.

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

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

  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 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 “Configuring Applications and Servers Dynamically” on page 5-1 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 can update this file manually and add or remove server configuration elements.

In this release of Oracle CEP, the `config.xml` file configures:

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

  See “Configuring Logging and Debugging” on page 11-1.

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

  See “Configuring Jetty for Oracle Complex Event Processing” on page 7-1.

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

  See “Configuring Access to a Relational Database” on page 9-1.

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

  See “Configuring JMX for Oracle Complex Event Processing” on page 8-1.

- HTTP publish-subscribe server.

  See “Configuring the HTTP Publish-Subscribe Server” on page 10-1.

The following sample `config.xml`, from the *`BEA_HOME`*`/user_projects/domains/wlevs30_domain/defaultserver` template domain, shows how to configure some of these services:

```
<?xml version="1.0" encoding="UTF-8"?>
<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>4098</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>1099</rmi-registry-port>
  </jmx>

  <ssl>
    <name>sslConfig</name>
    <key-store>./ssl/dsidentity.jks</key-store>
    <key-store-pass>
        <password>changeit</password>
    </key-store-pass>
    <key-store-alias>ds</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>pubsubbean</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>

</n1:config>
```

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 Clustered Domains

This section contains information on the following subjects:

Overview of Clustering and High Availability in Oracle Complex Event Processing

The following sections introduce use cases and terminology related to the management and availability of a set of Oracle Complex Event Processing (or *Oracle CEP* for short) instances.

WARNING: Clustering, as well as clustering terminology, in Oracle CEP is different to that of Oracle WebLogic Server.

Oracle CEP Clustered Domain

An Oracle CEP clustered domain is a set of servers logically connected for the purposes of management, and physically connected using a shared User Datagram Protocol (UDP) multicast address. All servers in an Oracle CEP clustered 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 clustering infrastructure is done at the domain level. Thus server failure, start, or restart is detected by every member of the clustered domain. Each member of the clustered domain has a consistent, agreed notion of domain membership enforced by the clustering infrastructure.

For servers to be considered a part of the same clustered domain they must share both the same multicast address and the same domain name.

Groups

In order to support the deployment to and management of the servers clustered domains 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, you deploy it to a particular group. Applications deployed to the domain or custom groups must have a unique name across the domain.

Cluster Notifications and Messaging

In order to provide high availibility (HA)-like capabilities to adapter 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.

Creating and Configuring a Simple Clustered Domain

This section describes how to create and configure a simple clustered domain from two or more Oracle CEP servers. The cluster 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 Clustered Domain" on page 4-5 for a description of how to configure custom groups within the clustered domain.

Note: It is assumed that you have already created the servers and you now want to create a clustered domain of these servers. For details on creating a server, see "Creating and Updating an Oracle CEP Domain" on page 2-1. Additionally, the following sections describe how to update the `config.xml` file for each server; this file is located in the `servername/config` directory of the main domain directory.

To create a simple clustered domain, update the `config.xml` file for each member server by adding a `<cluster>` child element of the root `<config>` element. Include the `<name>`, `<multicast-address>`, and `<identity>` child elements of `<cluster>`, as shown in the following example:

```
<config>

  <domain>
        <name>myDomain</name>
       <server>
           <name>myServer1</name>
       </server>
  </domain>

  <cluster>
      <name>myCluster</name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>1</identity>
      <enabled>true</enabled>
  </cluster>

...

</config>
```

In the example, the server called `myServer1` is part of the domain `myDomain` and also part of the clustered domain `myCluster`. By default, clustering is disabled, so you must explicitly enable it with the `<enabled>` element.

For each server of the clustered domain, the `<name>` and `<multicast-address>` elements must contain the same value. The `<identity>` element, however, must be different for each server in the clustered domain.

The `<multicast-address>` element is required unless all servers of the clustered 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 clustered 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 cluster operations; the server with the lowest identity becomes the domain coordinator. Be sure that each server in the clustered domain has a different identity; if servers have the same identity, the results of cluster operations are unpredictable.

An example of configuring a second server, called `myServer2`, in the `myCluster` clustered domain is shown below; note that its identity is 2:

```
<config>

  <domain>
        <name>myDomain</name>
       <server>
           <name>myServer2</name>
       </server>
  </domain>

  <cluster>
      <name>myCluster</name>
      <multicast-address>239.255.0.1</multicast-address>
      <identity>2</identity>
      <enabled>true</enabled>
  </cluster>

...

</config>
```

See "Additional Child Elements of the <cluster> Element" on page 4-12 for a brief description of additional cluster-related configuration elements.

Configuring Custom Groups for a Clustered Domain

There are cases where the application logic cannot simply be replicated across a homogenous set of clustered servers. 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 `server1` to be a member of the `selector` group and `server2` and `server3` 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>
```

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

Listing 4-2 Config.xml of myServer2

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

Listing 4-3 Config.xml of myServer3

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

Starting the Servers in a Clustered Domain

To start the servers in a clustered domain, start each server separately by running its start script. This is the same way you start a server even if it's not part of a clustered domain. See "Stopping and Starting the Server" on page 2-15 for details.

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

If, however, you have configured custom groups for the clustered 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 Clustered Domain

When you deploy an application to a clustered 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 deplyments to the new server.

Take, for example, the simple cluster configured in the section "Creating and Configuring a Simple Clustered Domain" on page 4-3. 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 clustered domain has been started, the application will be deployed only to `myServer1`. When `myServer2` is subsequently started, Oracle CEP *automatically* propagates the deployment of application to `myServer2` without the user having the explicitly deploy it.

Deployment propagation only occurs when a server is missing a required deployments. 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 clustered 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 clustered domains.

Note: When you upgrade a 2.0 domain to execute in a clustered 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 clustered 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 clustered domain has been configured as described in "Configuring Custom Groups for a Clustered Domain" on page 4-5.

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 cluster 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 -group selectorgroup
```

Based on the cluster 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 Cluster 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 cluster 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 paramter, `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.

Securing Cluster Messages

The servers in a clustered domain update their state by exchanging cluster-related messages. It is important that these messages be at least checked for integrity. A private key can be used to achieve integrity. This key must be shared by all servers in the domain.

To configure security for a clustered domain, add the `<security>` child element to the `<cluster>` element for each server in the clustered domain, as shown:

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

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

- `none`—Default value. Specifies that no security is configured for the clustered domain.
- `sign`—Specifies that cluster messages should be digitally signed.
- `encrypt`—Specifies that cluster messages should be encrypted.

To generate a key for digitally signing of cluster messages use:

```
genMSAKey -f <filename>
```

To generate a key for encrypting of cluster messages use:

```
No documentation available for Beta.
```

After you have generated the key for signing or encrypting, whichever is appropriate, put the key in the file called `.msainternal.dat`, located in the *`servername`* directory of the main domain directory; this file is generated when the server first starts up.

You must copy these same keys to each server in the domain.

Additional Child Elements of the <cluster> Element

The sections "Creating and Configuring a Simple Clustered Domain" on page 4-3 and "Configuring Custom Groups for a Clustered Domain" on page 4-5 and "Securing Cluster Messages" on page 4-11 discuss some of the child elements of the `<cluster>` element of the `config.xml` file, in particular `<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 clustered domain:

- `<server-name>`—Specifies a name for the server. Visualizer uses the value of this element when it displays the server in its console. Default value if element is not set is `Server-<identity>`.
- `<server-host-name>`—Specifies the host address/IP used for point-to-point HTTP cluster communication. Default value is `localhost`.
- `<multicast-port>`—Specifies the port used for multicast traffic. Default value is `9100`.
- `<operation-timeout>`—Specifies, in milliseconds, the timeout for point-to-point HTTP clustering requests. Default value is `30000`.

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. 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 8-1.
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 *`BEA_HOME`*`/wlevs30/bin/com.bea.wlevs.management.commandline_3.0.0.0.jar` file, where *`BEA_HOME`* refers to the main Oracle Home directory into which you installed Oracle CEP. 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:/beahome/wlevs30/bin/com.bea.wlevs.management.commandline_3.0.0.0.
     jar
     -url service:jmx:rmi:///jndi/rmi://localhost:1099/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:

- *BEA_HOME*/wlevs30/bin/com.bea.wlevs.management.commandline_3.0.0.0.jar
- *BEA_HOME*/wlevs30/bin/wlevs_3.0.jar
- *BEA_HOME*/wlevs30/modules/com.bea.wlevs.deployment.server_3.0.0.0.jar
- *BEA_HOME*/wlevs30/modules/com.bea.wlevs.ede_3.0.0.0.jar
- *BEA_HOME*/wlevs30/modules/com.bea.wlevs.management_3.0.0.0.jar
- *BEA_HOME*/modules/com.bea.core.jmx_6.0.0.0.jar
- *BEA_HOME*/modules/com.bea.core.jndi.context_6.0.0.0.jar
- *BEA_HOME*/modules/com.bea.core.rmi_6.0.0.0.jar
- *BEA_HOME*/modules/com.bea.core.i18n_1.4.0.0.jar
- *BEA_HOME*/modules/com.bea.core.diagnostics.core_2.1.0.0.jar
- *BEA_HOME*/modules/javax.xml.stream_1.1.1.0.jar

where *BEA_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:
   - *BEA_HOME*/wlevs30/bin/com.bea.wlevs.management.commandline_3.0.0.0.jar
   - *BEA_HOME*/wlevs30/modules/com.bea.wlevs.deployment.server_3.0.0.0.jar
   - *BEA_HOME*/wlevs30/modules/com.bea.wlevs.ede_3.0.0.0.jar
   - *BEA_HOME*/wlevs30/modules/com.bea.wlevs.management_3.0.0.0.jar
   - *BEA_HOME*/modules/com.bea.core.jmx_6.0.0.0.jar
   - *BEA_HOME*/modules/com.bea.core.jndi.context_6.0.0.0.jar
   - *BEA_HOME*/modules/com.bea.core.rmi_6.0.0.0.jar
   - *BEA_HOME*/modules/com.bea.core.i18n_1.4.0.0.jar

  - *BEA_HOME*/modules/com.bea.core.diagnostics.core_2.1.0.0.jar
  - *BEA_HOME*/modules/javax.xml.stream_1.1.1.0.jar

  where *BEA_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*/com.bea.wlevs.management.commandline_3.0.0.0.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.

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 5-5
- “User Credentials Arguments” on page 5-6
- “Common Arguments” on page 5-7

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 5-9

- “Commands for Managing the EPL Rules of an Application” on page 5-10
- “Commands for Managing Oracle CEP MBeans” on page 5-18

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 `1099`.
- 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.

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} ]
     [ 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>1099</rmi-registry-port>**`
	`  <rmi-jrmp-port>9999</rmi-jrmp-port>`
	`</jmx>`
	In the example, the JMX port is `1099`.
	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 1099, then the URL would be:
	`-url service:jmx:rmi:///jndi/rmi://ariel:1099/jmxrmi`
	See "Configuring JMX for Oracle Complex Event Processing" on page 8-1 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:1099/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 5-5 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:1099/jmxrmi
       -username wlevs -password wlevs
        SHUTDOWN -scheduleAt 600
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

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.
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.

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 5-25 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 5-25 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 rules 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 deployed to the specified Oracle CEP application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:1099/jmxrmi
        -username wlevs -password wlevs
        ADDRULE –application helloworld –processor helloworldProcessor
        –rule myrule "SELECT * FROM Withdrawal RETAIN 5 EVENTS"
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

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 5-25 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 5-25 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 5-25 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 deployed to the specified Oracle CEP application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:1099/jmxrmi
        -username wlevs -password wlevs
        DELETERULE –application helloworld –processor helloworldProcessor
        –rule myrule
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

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 5-9 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 5-25 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 5-25 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 5-25 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 deployed to the specified Oracle CEP application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:1099/jmxrmi
        -username wlevs -password wlevs
        GETRULE –application helloworld –processor helloworldProcessor
        –rule myrule
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

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 5-10 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 5-25 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 5-25 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 deployed to the specified Oracle CEP application:

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:1099/jmxrmi
        -username wlevs -password wlevs
        UPLOAD –application helloworld –processor helloworldProcessor
        –sourceURL file:///c:/processor/config/myrules.xml
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

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 5-15.

Syntax

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

Table 5-11 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 5-25 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 5-25 for details on getting the exact name if you do not know it.

Table 5-11 DOWNLOAD Arguments

Argument	Definition
`-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:1099/jmxrmi
        -username wlevs -password wlevs
        DOWNLOAD –application helloworld –processor helloworldProcessor
        –file c:\processor\config\myrules.xml
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

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 5-12 is an overview of the MBean management commands.

Table 5-12 MBean Management Command Overview

Command	Description
GET	Displays properties of MBeans.
INVOKE	Invokes management operations that an MBean exposes for its underlying resource.
QUERY	Searches for MBeans whose `ObjectName` matches a pattern that you specify.
SET	Sets the specified property values for the named MBean instance.

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 5-13 GET Arguments

Argument	Definition
`-type mbeanType`	Returns information for all MBeans of the specified type. For more information, see "Specifying MBean Types" on page 5-18.
`-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:1099/jmxrmi
         -username wlevs -password wlevs
         GET -pretty
         -mbean
com.bea.wlevs:Name=eplprocessor,Type=EPLProcessor,Application=helloworld
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

For more information about the environment in which this example runs, see "Example Environment" on page 5-5.

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

```
prompt> java wlevs.Admin
        -url service:jmx:rmi:///jndi/rmi://ariel:1099/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 5-14 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 5-18.
`-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:1099/jmxrmi
        -username wlevs -password wlevs
        INVOKE -mbean
com.bea.wlevs:Name=eplprocessor,Type=EPLProcessor,Application=helloworld
        -method addRule "SELECT * FROM Withdrawal RETAIN ALL"
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

For more information about the environment in which this example runs, see "Example Environment" on page 5-5.

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 5-15 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:1099/jmxrmi
        -username wlevs -password wlevs
         QUERY -pattern *:Type=EPLProcessor,*
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

If the command succeeds, it returns the following:

```
Ok
```

For more information about the environment in which this example runs, see "Example Environment" on page 5-5.

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:1099/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 5-16 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 5-18.
`-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 5-16 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:1099/jmxrmi
        -username wlevs -password wlevs
        SET -mbean
com.bea.wlevs:Name=helloworldOutstream,Type=Stream,Application=helloworld
        -property  MaxSize 1024
```

Note: For clarity, the preceding example is shown on multiple lines; however, when you run the command, enter all arguments and commands on a single line.

For more information about the environment in which this example runs, see "Example Environment" on page 5-5.

CHAPTER 6

Configuring Security for Oracle Complex Event Processing

This section contains information on the following subjects:

Overview of Security in Complex Event Processing

Oracle Complex Event Processing (or *Oracle CEP* for short) security can be configured to use one of several types of security providers. As initially installed, it is configured to use the file-based providers for authentication and authorization, with an administrator user with username `wlevs` and password `wlevs`. You can add your own users to this configuration, or you can configure the system to authenticate users from an LDAP or DBMS provider. The file-based authorization provider gives you a pre-configured mapping of roles to permissions. You can use one of the other authorization providers to create your own mappings instead.

Configuring Security With the File-Based Provider

The type of file-based provider security you can enable for Oracle CEP is basic authentication. Authentication is a process whereby the identity of users is proved or verified. Authentication typically involves username/password combinations.

All Oracle CEP examples and domains are configured to have an administrator with username `wlevs` and password `wlevs`.

By default, security is disabled in the HelloWorld example and in the template user domain located in the *`BEA_HOME`*`/user_projects/domains/wlevs30_domain` directory. 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.

The following table describes the available Oracle CEP security roles, as well as the name of the groups that are assigned to these roles.

Table 6-1 Available Oracle CEP Roles and Groups

Role	Description	Associated Group Name
Admin	Oracle CEP Administrator. Can perform all Oracle CEP tasks: start a server instance, deploy applications, and execute all commands of the `wlevs.Admin` utility. By default, all domains and examples include an administrator called `wlevs` with password `wlevs`.	`wlevsAdministrators`
Monitor	Can perform only monitoring tasks, such as the read-only commands of the `wlevs.Admin` utility.	`wlevsMonitors`

Configuring File-Based Security: Main Steps

The following procedure describes the steps to enable file-based provider security for your Oracle CEP domain. The procedure also shows how to add two users, one in the Admin role and the other in the Monitor role; this step is not required if the pre-configured `wlevs` username (with password `wlevs`) is adequate. For clarity, it is assumed that:

- The new user in the Admin role is called `jane` with password `secret`.
- The new user in the Monitor role is called `bob` with password `supersecret`.

To configure security:

1. Open a command window and set your environment as described in Setting Up Your Development Environment.
2. Add the *WLEVS_HOME*/bin directory to your PATH environment variable, where *WLEVS_HOME* is the main Oracle CEP installation directory, such as d:\beahome\wlevs30:

   ```
   prompt> set PATH=d:\beahome\wlevs30\bin;%PATH%  (Windows)
   prompt> PATH=/beahome/wlevs30/bin:$PATH (UNIX)
   ```

3. Change to the *DOMAIN_DIR*/config directory, where *DOMAIN_DIR* refers to the main directory of your domain, such as d:\beahome\user_projects\wlevs30_domain

   ```
   prompt> cd d:\beahome\user_projects\domains\wlevs30_domain
   ```

4. Execute the secgen.cmd (Windows) or secgen.sh (UNIX) script to generate a unique key for your installation:

   ```
   prompt> secgen -k -o security-key.dat
   ```

   See "The secgen Command Line Utility" on page 6-10 for additional information.

5. Create a new security configuration file that uses the new key by executing the following secgen command:

   ```
   prompt> secgen -F -o security.xml
   ```

   See "The secgen Command Line Utility" on page 6-10 for additional information.

6. Execute the passgen.cmd (Windows) or passgen.sh (UNIX) script to generate encrypted passwords for the new users; pass as the single argument the cleartext password. Use the output of the script to copy and paste the encrypted password.

   ```
   prompt> passgen.cmd secret
   {SHA-1}C19+kCHEcB1ZQ/ZjuiQxNsNbFGoQZYw=
   prompt> passgen.cmd supersecret
   {SHA-1}LoLDn/H1aC2qSoExZ+yNBnl1Tbqys/8=
   ```

   See "The passgen Command Line Utility" on page 6-7 for additional information.

7. Using your favorite text editor, edit the file atnstore.txt following these guidelines:
   - For each new user you want to configure, add two lines at the beginning of the file. Start each line with the user: and password: keywords, then enter the value. Be sure you enter the encrypted password you generated in a preceding step.

     You must also add a blank line between each user entry, as shown in the example below.

– To assign a user to the Admin role, add a `member: user` entry below the `group: wlevsAdministrators` entry, where *`user`* refers to the username.

– To assign a user to the Monitor role, add a `member: user` entry below the `group: wlevsMonitors` entry, where *`user`* refers to the username.

The following sample `atnstore.txt` file shows configurations for the `wlevs`, `jane`, and `bob` users:

```
user: wlevs
password: {SHA-1}8MHYCQlzyh/S7TAbEC+fU34o4rf7GVM=


user: jane
password: {SHA-1}C19+kCHEcB1ZQ/ZjuiQxNsNbFGoQZYw=


user: bob
password: {SHA-1}LoLDn/H1aC2qSoExZ+yNBnl1Tbqys/8=

group: wlevsAdministrators
member: wlevs
member: jane

group: wlevsMonitors
member: bob
```

8. Update the `startwlevs.cmd` (Windows) or `startwlevs.sh` (UNIX) start script by adding the following two arguments to the `java` command that starts Oracle CEP:

   ```
   -Dcom.bea.core.security.username=username
   -Dcom.bea.core.security.password=password
   ```

   where *`username`* and *`password`* refer to the administrator user you have configured for Oracle CEP, `wlevs` by default. The server start scripts are located in the main directory of your domain, such as *`BEA_HOME`*`/user_projects/domains/wlevs30_domain`.

   The following snippet from the Windows `startwlevs.cmd` script shows in bold how to specify the administrator username and password:

   ```
    %JAVA_HOME%\bin\java -Dwlevs.home=%USER_INSTALL_DIR%
   -Dbea.home=%BEA_HOME% -Dcom.bea.core.security.username=jane
   -Dcom.bea.core.security.password=secret -jar
   "..\..\..\bin\wlevs_3.0.jar" %1 %2 %3 %4 %5 %6
   ```

   If you do not want to put cleartext passwords in the `startwlevs.cmd` script, see "Avoiding Cleartext Passwords in the startwlevs Script" on page 6-5.

9. Restart Oracle CEP for the security to take effect. See "Stopping and Starting the Server" on page 2-15.

After you have enabled security, you must use the `-username` and `-password` arguments to the Deployer and `wlevs.Admin` tool to specify the administrator user.

Avoiding Cleartext Passwords in the startwlevs Script

The procedure in "Configuring File-Based Security: Main Steps" on page 6-2 describes how to update the `startwlevs` server start script with the cleartext password of the administrator user. Cleartext passwords, however, are typically not allowed in a production environment. There are two ways to not use cleartext passwords when starting the server:

- "Being Prompted for a Password at Server Startup" on page 6-5
- "Putting the Encrypted Password in a Security XML file" on page 6-5

Being Prompted for a Password at Server Startup

If you want the server start script to prompt you for the administrator password, then update the script by adding the following argument to the `java` command that starts Oracle CEP:

```
-Dcom.bea.core.security.prompt=true
```

In this case do *not* specify the `com.bea.core.security.username` or `com.bea.core.security.password` arguments in the script.

When the script prompts you for a password, the password field will not be echoed to the screen on platforms where this is supported.

The following snippet from the Windows `startwlevs.cmd` script shows in bold how to specify that you be prompted for a password when starting the server:

```
 %JAVA_HOME%\bin\java -Dwlevs.home=%USER_INSTALL_DIR%
-Dbea.home=%BEA_HOME% -Dcom.bea.core.security.prompt=true -jar
"..\..\..\bin\wlevs_3.0.jar" %1 %2 %3 %4 %5 %6
```

Putting the Encrypted Password in a Security XML file

Follow these steps if you want to put the encrypted administrator password in an XML file that will then be accessed by the server start script.

1. Open a command window and set your environment as described in Setting Up Your Development Environment.
2. Change to the *`DOMAIN_DIR`*`/config` directory, where *`DOMAIN_DIR`* refers to the main directory of your domain, such as `d:\beahome\user_projects\wlevs30_domain`

3. Using your favorite XML editor, edit the `security-config.xml` file by adding an `<msa-security>` child element to the `<config>` root element. Use the two child elements of `<msa-security>` shown below to specify the administrator's username and cleartext password:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <css-realm>
  ...
  <msa-security>
    <boot-user-name-encrypted>jane</boot-user-name-encrypted>
    <password>secret</password>
  </msa-security>
</config>
```

4. Execute the following `java` command to encrypt the cleartext password in the `security-config.xml` file:

```
prompt> java -jar BEA_HOME/modules/com.bea.core.bootbundle_3.0.1.0.jar .
security-config.xml
```

   where *`BEA_HOME`* refers to the main directory into which you installed Oracle CEP, such as `d:\beahome`. The second argument refers to the directory that contains the `security-config.xml` file; because this procedure directs you to change to the directory, the example shows `"."`.

   After you run the command, the `security-config.xml` file will use encrypted passwords, such as the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <css-realm>
  ...
  <msa-security>

<boot-user-name-encrypted>{Salted-3DES}dBcBdaoemdY=</boot-user-name-enc
rypted>

    <password>{Salted-3DES}X86x/+El2/s=</password>
  </msa-security>
</config>
```

5. Copy the file `.msaInternal.dat`, generated by the `java` command in the preceding step, from the directory in which you ran the `java` command to the main domain directory. For example:

```
prompt> copy
d:\beahome\user_projects\wlevs30_domain\config\.msaInternal.dat
d:\beahome\user_projects\wlevs30_domain
```

Disabling File-Based Security

To disable file-based 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 main directory of your domain, such as *`BEA_HOME`*`/user_projects/domains/wlevs30_domain`.

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

```
%JAVA_HOME%\bin\java -Dwlevs.home=%USER_INSTALL_DIR% -Dbea.home=%BEA_HOME% -jar
"%USER_INSTALL_DIR%\bin\wlevs_3.0.jar" -disablesecurity %1 %2 %3 %4 %5 %6
```

The passgen Command Line Utility

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

The `passgen` utility is located in the *`WLEVS_HOME`*`/bin` directory, where *`WLEVS_HOME`* is the main Oracle CEP installation directory, such as `d:\beahome\wlevs30`. The utility comes in two flavors:

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

passgen Syntax

```
prompt> 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 6-9
- "Providing a Password on the Command Line" on page 6-9

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

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

The `secgen` utility is located in the *`WLEVS_HOME`*`/bin` directory, where *`WLEVS_HOME`* is the main Oracle CEP installation directory, such as `d:\beahome\wlevs30`. 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.

```
prompt> 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`.
`-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.

Option	Description	Comments
`-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 6-11 for details.	A `SecGenTemplate.properties` template file is located at *`WLEVS_HOME`*`/bin` where *`WLEVS_HOME`* is the main installation directory of Oracle CEP, such as `/beahome/wlevs30`.

Generating a Key File

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

```
prompt> 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 *`WLEVS_HOME`*`/bin` where *`WLEVS_HOME`* is the main installation directory of Oracle CEP, such as `/beahome/wlevs30`.

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 7

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 7-4.

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 7-9.

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 7-4 for details.
- `<netio>`: See "netio Configuration Object" on page 7-5 for details.
- `<work-manager>`: See "work-manager Configuration Object" on page 7-6 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 7-6 for details.

See "Example Jetty Configuration" on page 7-8 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 7-1 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 7-5 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 7-6.
`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 7-2 Configuration Parameters for <netio>

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

Table 7-2 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 7-3 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 7-4 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 7-4. 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 7-4.
`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 7-6.

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 7-1 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 8

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 8-2 for details.
- `<rmi>`: See "rmi Configuration Object" on page 8-2 for details.
- `<jndi-context>`: See "jndi-context Configuration Object" on page 8-3 for details.
- `<exported-jndi-context>`: See "exported-jndi-context Configuration Object" on page 8-4 for details

See "Example of Configuring JMX" on page 8-5 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 8-1 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 8-2 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 7-1).
`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 8-3 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 8-3.

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 8-4 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 8-2.
`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 8-1

```
<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 9

Configuring Access to a Relational Database

This section contains information on the following subjects:

Overview of Database Access from an Oracle Complex Event Processing 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.

Oracle CEP provides the following JDBC drivers, installed in the *`WLEVS_HOME`*`/bin` directory, where *`WLEVS_HOME`* refers to the main installation directory such as `d:\beahome\wlevs30`.

- Oracle 10.2.0 thin driver (packaged in the `com.bea.oracle.ojdbc14_10.2.0.jar` JAR file)

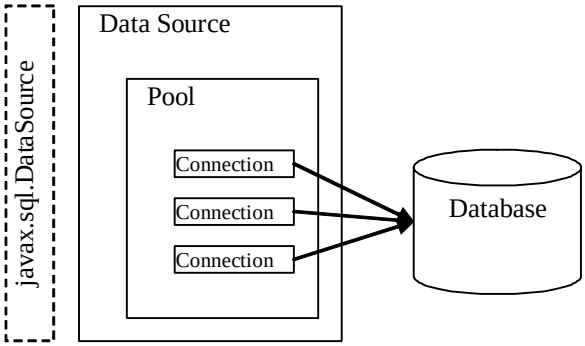
- Microsoft SQL (packaged in the `com.bea.microsoft.sqljdbc_1.0.jar` JAR file)

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

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 9-1 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. Following is a sample datasource `config.xml` section that illustrates the configuration format.

```
<data-source>
  <name>oraxads2</name>
  <driver-params>
```

```
      <url>jdbc:oracle:thin:@buckhorn.bea.com:1521:ce102a</url>
      <driver-name>oracle.jdbc.xa.client.OracleXADataSource</driver-name>
      <properties>
        <element>
          <name>user</name>
          <value>cedeployqa</value>
        </element>
        <element>
          <name>password</name>
          <value>cedeployqa123</value>
        </element>
      </properties>
      <use-xa-data-source-interface>true</use-xa-data-source-interface>
    </driver-params>
    <connection-pool-params>
      <initial-capacity>15</initial-capacity>
      <max-capacity>50</max-capacity>
      <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
    </connection-pool-params>
    <data-source-params>
      <jndi-names>
        <element>oraxads2</element>
      </jndi-names>

<global-transactions-protocol>TwoPhaseCommit</global-transactions-protocol
>
    </data-source-params>
  </data-source>

  <transaction-manager>
    <name>tm1</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 8-2.

Configuring Access to a Relational Database: Main Steps

Follow these steps to configure and use JDBC in your application:

1. Update the server start script in 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 main domain directory. The out-of-the-box sample domains are located in *`WLEVS_HOME`*`/samples/domains`, and the user domains are located in *`BEA_HOME`*`/user_projects/domains`, where *`WLEVS_HOME`* refers to the main Oracle CEP installation directory, such as `d:\beahome\wlevs30`, and *`BEA_HOME`* refers to the directory above *`WLEVS_HOME`*, such as `d:\beahome`.

   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 *`WLEVS_HOME`*.

2. If are configuring JDBC for use in an EPL rule, or you want to use the Oracle CEP `DataSource`, configure the data source in the server's `config.xml` file.

   For details, see "Data Source Configuration" on page 9-2.

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

CHAPTER 10

Configuring the HTTP Publish-Subscribe Server

This section contains information on the following subjects:

Overview of HTTP Publish-Subscribe Server

No documentation available for Beta.

Configuring the HTTP Publish-Subscribe Server

No documentation available for Beta.

Example of Configuring the HTTP Publish-Subscribe Server

No documentation available for Beta.

CHAPTER 11

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.

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 an `Oracle.logging.commons.LogFactoryImpl`.

- "Setting the Log Factory" on page 11-2
- "Using Log Severity Levels" on page 11-3
- "Log Message Format" on page 11-4
- "OSGI Framework Logger" on page 11-5

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= an
  Oracle.logging.commons.LogFactoryImpl
  ```

- You can programmatically implement the logging by:

  ```
  import org.apache.commons.logging.LogFactory;

  System.setProperty(LogFactory.FACTORY_PROPERTY, "an
  Oracle.logging.commons.LogFactoryImpl");
  ```

- The `an Oracle.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 and the handler. 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.

Table 11-1 lists the severity levels of log messages.

Table 11-1 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.

Table 11-1 Message Severity (Continued)

Severity	Meaning
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 an `Oracle.logging.commons.LogFactoryImpl`.

   This `LogFactory` creates instances of an `Oracle.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 an `Oracle.logging.commons.LogImpl` log methods direct the message to the server log.

Listing 11-1 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,
      "an Oracle.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 11-2 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 11-8.
`stdout-config`	`String`	The name of the `stdout` configuration object used to configure `stdout` output. See "log-stdout" on page 11-7.
`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`.
`name`	`String`	The name of this configuration object.

log-stdout

This section provides information on the `log-stdout` configuration object:

Table 11-3 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`.
`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 11-4 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
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.

Table 11-4 Configuration Parameters for log-file

Parameter	Type	Description
`log-file-severity`	`String`	Specifies the least important severity of messages written to the log file. Default value is `Trace`. Valid values are: `Emergency`, `Alert`, `Critical`, `Error`, `Warning`, `Notice`, `Info`, `Debug`, `Trace`.
`log-file-rotation-dir`	`String`	Specifies the directory where old rotated files are stored. If not set, the old files are stored in the same directory as the base log file.
`name`	`String`	The name of this configuration object.

Debug

The following sections provide information on how to use the Oracle CEP debugging feature:

- "Configuring debug using System Properties" on page 11-10
- "Configuring debug using a Configuration File" on page 11-11
- "Supported Debug Flags" on page 11-11
- "Example Debug Configuration" on page 11-14

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 11-5 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
`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

Table 11-5 Debug Flags

Debug Flag	Description
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
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

Table 11-5 Debug Flags

Debug Flag	Description
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
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

Table 11-5 Debug Flags

Debug Flag	Description
`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 11-2 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:

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 11-3 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.