Oracle® Fusion Middleware

Getting Started Guide for Oracle Event Processing for Oracle Java Embedded

11*g* Release 1 (11.1.1.7)

E39318-01

February 2013

Documentation for administrators and developers that describes how to get started with Oracle Event Processing for Oracle Java Embedded, a Java server for developing high-performance embedded event-driven applications. It includes an overview of features and concepts and installation guidelines.

Oracle Fusion Middleware Getting Started Guide for Oracle Event Processing for Oracle Java Embedded 11*g* Release 1 (11.1.1.7)

E39318-01

Primary Author: Oracle Corporation

Contents

List of Tables

Preface

This document provides introductory information about Oracle Event Processing for Oracle Java Embedded.

Audience

This document is intended for users interested in learning about Oracle Event Processing for Oracle Java Embedded. Readers should be familiar with basic Java development. Some knowledge of SQL would be helpful.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Event Processing documentation set:

- *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*
- *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*
- *Oracle Fusion Middleware Java API Reference for Oracle Event Processing for Oracle Java Embedded*
- *Oracle Fusion Middleware CQL Language Reference for Oracle Event Processing*
- Oracle Event Processing Forum: `http://forums.oracle.com/forums/forum.jspa?forumID=820`

Conventions

The following text conventions are used in this document:

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

What's New in This Guide

This guide has been updated in several ways. The following table lists the sections that have been added or changed.

For a list of known issues (release notes), see the "Known Issues for for Oracle SOA Products and Oracle AIA Foundation Pack" at http://www.oracle.com/technetwork/middleware/docs/soa-aiafp-knownissuesindex-364630.html.

Sections	Changes Made	February 2013
Entire Guide	Guide created.	X

1 Overview of Oracle Event Processing for Oracle Java Embedded

This chapter provides an overview of Oracle Event Processing for Oracle Java Embedded, an event processing server designed for use in embedded environments. It introduces features and use cases specific to the embedded release, providing links to more information in Oracle Event Processing documentation.

This guide introduces Oracle Event Processing for Oracle Java Embedded and provides installation information. Note that Oracle Event Processing for Oracle Java Embedded features represent a subset of Oracle Event Processing features. For more about Oracle Event Processing, be sure to see *Oracle Fusion Middleware Getting Started Guide for Oracle Event Processing*.

This chapter includes the following sections:

1.1 Introduction to Oracle Event Processing for Oracle Java Embedded

Oracle Event Processing for Oracle Java Embedded is a server to support event processing applications in embedded environments, such as those supported by the Java Embedded Suite (JES).

Event processing applications receive potentially large amounts of streaming data representing events, then respond in real time based on the event data. Oracle Event Processing for Oracle Java Embedded is designed to support applications deployed in embedded environments often found at or near event sources. These environments include sensors such as for environment conditions and moving sources such as vehicles or mobile devices.

By deploying event processing applications in these locations, you can filter events at or near the event source, reducing the amount of network traffic flowing through other server resources, including Oracle Event Processing applications on enterprise servers.

Oracle Event Processing for Oracle Java Embedded includes a subset of the functionality available in Oracle Event Processing. It also includes functionality specific to embedded environments. For a list of features both included and omitted, see Section 1.1.2, "Features Included in Oracle Event Processing for Oracle Java Embedded".

1.1.1 Oracle Event Processing for Oracle Java Embedded Use Cases

The use cases described in this section illustrate specific uses for Oracle Event Processing for Oracle Java Embedded applications.

Temperature Analysis

A company provides "smart home" services, in part by monitoring temperature events from residential thermostats. The company wants finer control over event analysis and to reduce network traffic.

The company receives temperature events over the Internet from thermostats in many locations. Data from the thermostats is used to identify patterns and possible alert conditions.

Within an Oracle Event Processing for Oracle Java Embedded application embedded in the thermostat devices, Oracle CQL queries aggregate the event data and do threshold analysis before the events are sent over the Internet. As a result, the events received have already been identified as worth attention.

Server Room Monitoring

A company offering data management devices and services needs to improve its data center coordination and energy management in order to reduce total cost of ownership. The company needs finer-grained, more detailed sensor and data center reporting.

The company receives energy usage sensor data from disparate resources. Data from each sensor must be analyzed for its local relevance, then must be aggregated with data from other sensors to identify patterns that can be used to improve efficiency.

Separate Oracle Event Processing applications provide a two-tiered approach. In one tier, sensors at the very edge of the network represent event sources, sending data to gateway devices. These devices run Java SE, Java Embedded Suite (JES), and Oracle Event Processing for Oracle Java Embedded. Oracle Event Processing for Oracle Java Embedded applications on the devices apply intelligence as Oracle Continuous Query Language (Oracle CQL) to filter events generated by the sensors. In this way, only event data meeting filter criteria is sent to backend servers in the datacenter. These backend servers represent the approach’s other tier. Through Oracle Event Processing applications, they send alerts when needed. They also aggregate and correlate data from across the system to identify consistency issues and produce data to be used in reports on patterns.

1.1.2 Features Included in Oracle Event Processing for Oracle Java Embedded

Oracle Event Processing for Oracle Java Embedded includes a subset of the features provided in Oracle Event Processing. This subset is designed to support the particular application needs of event processing in an embedded environment. This section lists the features that are included and those that are not.

Note that there are no tool-related warnings related to using non-included features in an embedded application. In other words, features not included in the Oracle Event Processing for Oracle Java Embedded server may still appear to be supported in the Eclipse IDE. When using the IDE to develop applications, take care to include in applications only those server features that are supported in Oracle Event Processing for Oracle Java Embedded.

Features included in the embedded release are the same as those you will find in Oracle Event Processing.

Table 1–1 Features Included in Oracle Event Processing for Oracle Java Embedded

Feature	Notes
Programming model	As with traditional Oracle Event Processing, you develop embedded applications as event processing networks (EPNs). Note that some features of the programming model are *not* supported, including features that support tuning for high availabiility and scalability. For more information, see "Oracle Event Processing Programming Model" in *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.
Oracle Continuous Query Language (CQL)	Oracle CQL is supported, with the exception of functionality provided by the cartridges listed in Table 1–2.
Java cartridge	The Java cartridge enhances Oracle CQL with the ability to invoke Java code in Oracle CQL code. For more information, see "Oracle Java Data Cartridge" in *Oracle Fusion Middleware CQL Language Reference for Oracle Event Processing*.
Data source access	This release supports access to Java DB via drivers included with Java Embedded Suite (JES). You can configure a datasource in the standard way using the config.xml file. For more information about database access, see Section 1.2, "Oracle Event Processing for Oracle Java Embedded Application and Development Framework".
RESTful web services	You can implement RESTful web services through the Jersey JAX-RS support included with Java Embedded Suite. For more information, see REST Web Services Support.
Local cache	You can use the included local caching service. Coherence caching is not supported in this release. For more about caching support, see Section 1.2, "Oracle Event Processing for Oracle Java Embedded Application and Development Framework".
Security	Including authentication and SSL and utilities such as policygen, cssconfig, and encryptMSAConfig. For more on cssconfig and encryptMSAConfig, see Security Utilities Command-Line Reference in *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*. For more on policygen, see Section 1.6.1, "The policygen Command-Line Utility".
Jetty (HTTPS) service	For more information on Jetty in Oracle Event Processing for Oracle Java Embedded, see Section 1.2, "Oracle Event Processing for Oracle Java Embedded Application and Development Framework".
Configuration	You can use the Configuration Wizard silent mode to create and configure domains. For more information, see Section 1.3, "Creating Domains and Configuring the Server".

Table 1–1 (Cont.) Features Included in Oracle Event Processing for Oracle Java

Feature	Notes
Deployer	You can deploy applications from the command line with the Deployer tool. For more information on Deployer support in Oracle Event Processing for Oracle Java Embedded, see Section 1.5, "Deploying Oracle Event Processing for Oracle Java Embedded Applications".
Admin tool	For more information on using wlevs.Admin, see Section 1.6, "Administering Oracle Event Processing for Oracle Java Embedded Applications"
Management through JMX	This release supports management through Java Management Extensions (JMX). For more about JMX in Oracle Event Processing for Oracle Java Embedded, see Section 1.6, "Administering Oracle Event Processing for Oracle Java Embedded Applications".
Logging	For more about logging, see "Configuring Logging and Debugging" for Oracle Event Processing in *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.

Table 1–2 lists the Oracle Event Processing features that are not supported in Oracle Event Processing for Oracle Java Embedded. If you attempt to deploy an application that uses these features, you may receive a deployment or runtime error or failure.

Note that it might be useful to use some features early in development, such as the load generator and CSV adapter, as long as support for these is removed before you deploy the application into an embedded environment.

Table 1–2 Features Not Included in Oracle Event Processing for Oracle Java Embedded

Feature	Notes
Clustering	Embedded applications cannot be clustered.
Coherence caching	Caching through the included local cache service is supported.
Oracle Event Processing Visualizer	Oracle Event Processing Visualizer is not available as a user interface for managing applications. You can manage applications using included command line tools, including Deployer and the Admin tool.
HTTP publish-subscribe server and adapter	This release does not include the HTTP publish-subscribe server.
JMS adapter and WebLogic Server JMS client API	This release does not include support for JMS.
CSV file adapter	This release does not include an adapter for using a CSV file as test event data.
Load generator	This release does not include the tool for generating event data.
Event recording and playback	This release does not include the ability to record event activity and play it back for testing and debugging.
JDBC drivers, including those for SQL Server, Oracle JDBC, and Berkeley DB	However, you can configure a datasource in the standard way using the config.xml file. This release supports access to Java DB via drivers included with Java Embedded Suite (JES).
Oracle CQL Cartridges (except Java)	Of the Oracle CQL cartridges, which provide support for Oracle CQL enhancements, only the Java cartridge is supported. In particular, the spatial and JDBC cartridges are *not* supported.

Table 1–2 (Cont.) Features Not Included in Oracle Event Processing for Oracle Java

Feature	Notes
Java Persistence API (JPA) for object-relational mapping	
Java Architecture for XML Binding (JAXB)	
SOAP web services	
Some security utilities	These include passgen, secgen, and GrabCert.
Monitoring	
Event inspect and trace	This release does not include features to view events flowing from EPN stages and inject events into EPN stages.

1.2 Oracle Event Processing for Oracle Java Embedded Application and Development Framework

Oracle Event Processing for Oracle Java Embedded is designed to support technologies included in the Java Embedded Suite (JES). These include the Java DB for database access, Jersey for RESTful web services support, and the JES JRE.

Note that you can use the JES monitoring tool, jconsole, remotely. Local connections aren't supported.

For JES documentation, see the Oracle web site at `http://www.oracle.com/technetwork/java/embedded/resources/java-embedded-suite/index.html`.

The following describes technologies that are part of the framework supporting Oracle Event Processing for Oracle Java Embedded.

Database Access

Oracle Event Processing for Oracle Java Embedded supports the Java DB database engine (based on the Derby project) included with JES. You can use Java DB in either client mode or embedded server mode. You enable access to a Java DB database by copying the driver to the correct location and configuring the data source in the server configuration file.

Note that Oracle Event Processing for Oracle Java Embedded itself does not include any JDBC drivers by default.

Table 1–3 lists Derby drivers and how you can get them:

Table 1–3 Location of Derby Drivers to Support Java DB Access

Driver	Where to Find It
Embedded Java DB driver	The embedded driver, derby.jar, is included with JES at the following JES installation directory: `JES_INSTALL_DIR/javadb/lib`
Java DB client driver (used for client access to a remote Java DB instance)	The client driver, derbyclient.jar, is not included with JES. You can download the correct version at the following URL: `http://db.apache.org/derby/releases/release-10.8.2.2.html`

Once you have the driver JAR file, copy it to the following location in the Oracle Event Processing for Oracle Java Embedded server file system:

```
DOMAIN_HOME\SERVER_HOME\modules\ext
```

With the driver JAR in the correct location, you can configure data sources through a `data-source` element you add to the server config.xml file.

For more information about configuring a data source, see information about the `data-source` element in "Schema Reference: Server Configuration wlevs_server_config.xsd" in *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

Web Server

Oracle Event Processing for Oracle Java Embedded includes the Jetty web server and supports the Java Servlet API. The standard `org.osgi.service.http.HttpService` interface is also supported, allowing servlets to be dynamically registered.

For more information about Jetty, see "Configuring Jetty for Oracle Event Processing" in *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.

Cache Support

As in Oracle Event Processing, in Oracle Event Processing for Oracle Java Embedded includes simple caching support. Using this local cache, you can write applications that access cached data for faster processing. Note that Oracle Coherence is not supported, as it is on Oracle Event Processing.

For more information about the local cache, see "Configuring an Oracle Event Processing Local Caching System and Cache" in *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*

REST Web Services Support

You can implement web services to expose aspects of your Oracle Event Processing for Oracle Java Embedded applications. Oracle Event Processing for Oracle Java Embedded supports your implementing RESTful web services based on the JAX-RS standard by using the Jersey libraries included with Java Embedded Suite. Jersey is the reference implementation of the Java API for RESTful Web Services (JAX-RS) standard.

For more on enabling REST support in your Oracle Event Processing for Oracle Java Embedded application, see Section 1.4.2, "Enabling Support for REST Web Services".

For more information about Jersey, be sure to see the Jersey User Guide at `http://jersey.java.net/nonav/documentation/latest/user-guide.html`.

1.3 Creating Domains and Configuring the Server

You can use the Configuration Wizard to create a new domain on which to deploy Oracle Event Processing for Oracle Java Embedded applications. Oracle Event Processing for Oracle Java Embedded supports the Configuration Wizard in silent mode, a non-interactive way to create and configure a domain.

The Configuration Wizard creates a single default server in the domain; all of the server-related files are located in a subdirectory of the domain directory named the same as the server.

Note: The public-key certificate you create must include the full host name as its `cn` name, while the certificate's alias name should be "evsidentity".

To run the Configuration Wizard using silent mode:

1. Create a silent.xml file that defines the domain configuration settings.

   Note that incorrect entries in the silent.xml file can cause failures. For information to determine the cause of a failure, create a log file when you launch the Configuration Wizard.

2. Open a command window and change to the ORACLE_OEP_HOME/ocep_11.1/common/bin directory, where ORACLE_OEP_HOME refers to the installation directory, such as /oracle_oep:

   ```
   cd /oracle_oep/ocep_11.1/common/bin
   ```

3. Invoke the config.sh shell command in silent mode:

   ```
   sh config.sh -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 a preceding step.

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

   ```
   sh config.sh -mode=silent -silent_xml=path_to_xml_file -log=/home/logs/create_
   domain.log
   ```

When it completes successfully, the command will return messages such as those in Example 1–1:

Example 1–1 Output on Successful Completion of Config Wizard

```
/home/testuser/java/jes7.0/jre/bin is added to path
<Tue Feb 26 01:41:23 PST 2013> <Info> <BootBundle> <BEA-1004030> <An encryption
key file at location /home/testuser/test_domain/test_server/.aesinternal.dat has
been generated>
testuser@emb-sca-ti-xm-2:~/Oracle/Middleware0220/ocep_11.1/common/bin$
```

If the config wizard does not complete successfully, check the log file for more information. In addition, the shell command exit code can help you learn more about the outcome of silent execution. For more information, see Configuration Wizard Silent Mode Exit Codes.

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. A sample silent.xml file is included with the installation at ORACLE_OEP_HOME/common/lib/silent.xml. You might find this useful as a starting place for creating your own.

To create a silent.xml file:

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.

   Example 1–2 illustrates a silent.xml file.

Example 1–2 silent.xml File Example

```
<?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_server" />
        <data-value name="DOMAIN_NAME" value="mydomain" />
        <data-value name="DOMAIN_LOCATION" value="/home/mydomains" />

        <data-value name="NETIO_PORT" value="9002" />
        <data-value name="KEYSTORE_PASSWORD" value="welcome1" />

        <data-value name="DB_URL" value="jdbc:oracle:thin:@hostname.com:1521:xe"
/>
        <data-value name="DB_USERNAME" value="wlevs" />
        <data-value name="DB_PASSWORD" value="wlevs" />
    </input-fields>
</bea-installer>
```

2. Copy the contents of the sample XML file into your own silent.xml file.
3. In the silent.xml file you just created, edit the values for the keywords shown in Table 1–4 to reflect your configuration.

   For example, if you want to create the new domain in the /home/oracle_oep/user_projects/domains directory, update the corresponding <data-value> element as follows

   ```
   <data-value name="DOMAIN_LOCATION"
       value="home/oracle_oep/user_projects/domains" />
   ```

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

Table 1–4 Data-value names for silent.xml file

For this data-value name...	Enter the following value...
CONFIGURATION_OPTION	Specifies whether you want to create a new domain with a default server or update a server in an existing domain. Valid values are `createDomain` or `updateDomain`. Default value is `createDomain`.
USERNAME	The username of the administrator of the created or updated server in the domain.
PASSWORD	The password of the administrator of the created or updated server in the domain.
SERVER_NAME	The name of the new server in this domain. This name will also be used as the name of the directory that contains the server files.
DOMAIN_NAME	The name of the domain.
DOMAIN_LOCATION	The full name of the directory that will contain the domain. The standard location for domains is ORACLE_OEP_HOME/user_projects/domains, where ORACLE_OEP_HOME refers to the top-level installation directory.
NETIO_PORT	The port number to which the server instance itself listens.

Table 1–4 (Cont.) Data-value names for silent.xml file

For this data-value name...	Enter the following value...
KEYSTORE_PASSWORD	The password for the identity keystore.
PRIVATEKEY_PASSWORD	The password for the certificate private key. The default value of this option is the value of the KEYSTORE_ PASSWORD.
DB_URL	The URL used to connect to a database using JDBC. This option is used to configure the data source. The database configuration parameters are optional; if you do not specify them, then no datasource is configured for the server. Note that the configuration wizard supports configuring a data source using the Oracle database only. To use another database technology, please edit the configuration manually.
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.

Configuration Wizard Silent Mode Exit Codes

When run in silent mode, the Configuration Wizard generates exit codes that indicate the success or failure of domain creation and configuration. These exit codes are shown in Table 1–5.

Table 1–5 Configuration Wizard Silent Mode 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

1.4 Developing Oracle Event Processing for Oracle Java Embedded Applications

You can develop Oracle Event Processing for Oracle Java Embedded applications using the Oracle Event Processing IDE for Eclipse. The IDE includes a visual editor for designing event processing networks (EPNs), as well as other tools that make developing applications easier.

Keep in mind, however, that the IDE currently includes no support for ensuring that applications you build will deploy and run in an embedded environment. For example, the IDE does not validate that the application you are designing does not include features that are not supported.

In other words, when developing embedded applications, you must take care to limit the set of features you use to those that are supported on Oracle Event Processing for Oracle Java Embedded. This is a subset of the features to which the IDE itself provides access.

If you create applications using features unsupported by Oracle Event Processing for Oracle Java Embedded (whether or not you use the IDE), you may see deployment-time errors or runtime failures.

For a list of Oracle Event Processing features included in (and excluded from) Oracle Event Processing for Oracle Java Embedded, see Section 1.1.2, "Features Included in Oracle Event Processing for Oracle Java Embedded".

For more information about developing event processing applications with the IDE, see Overview of Creating Oracle Event Processing Applications in *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

Note: It is possible, when starting the server, that you will see an error message in console output such as "Unable to load the native wlfileio library for the persistent file store...". This message is benign and can be ignored.

1.4.1 Oracle Event Processing for Oracle Java Embedded Sample

Oracle Event Processing for Oracle Java Embedded includes the Smart Appliance sample application. The application listens for device events via socket connection. The device events represent device information and the values associated with it. The threshold values for each device is stored in a JavaDB database. As the application processes device events, if an event's value does not fall within specified threshold limits, then the application generates the alerts.

For more information about the sample, including how to run it, be sure to see the readme.txt file included with it.

After you install Oracle Event Processing for Oracle Java Embedded, you will find the Smart Appliance application at the following file system location:

```
ORACLE_OEP_HOME/ocep_11.1/examples/source/smartappliance
```

1.4.2 Enabling Support for REST Web Services

You can expose RESTful web services from your application by using Jersey, the JAX-RS implementation included with Java Embedded Suite. Once you have REST support enabled, you can annotate methods of your Java code so that they execute when REST client calls are made. For more information about using Jersey and JAX-RS, be sure to see the Jersey User Guide at `http://jersey.java.net/nonav/documentation/latest/user-guide.html`.

Note that the Smart Appliance sample application is an example of a REST-enabled application. For more information, see Section 1.4.1, "Oracle Event Processing for Oracle Java Embedded Sample".

Enabling Support for REST Web Services

To enable REST support, you must ensure that the Jersey libraries are on your application's classpath so that you can use them in classes that implement methods as REST resources. You must also configure your RESTful classes with the OSGi service so that they are available to external clients.

To enable REST web services support, perform the following steps:

1. Ensure that the Jersey libraries are in your application's classpath.

For example, you might put the JAR files in the `modules/ext` directory of your Oracle Event Processing for Oracle Java Embedded server domain. Once the libraries are there, you will be able to import Jersey packages as needed into your own code and use the Jersey API as described in its documentation.

2. In your application's assembly XML file, include an OSGi HTTP Service reference and associate it with the bean that represents your RESTful class, as shown in Example 1–3:

Example 1–3 Referencing the OSGi HTTP Service in an Assembly XML File

```
<osgi:reference id="httpServiceRef" interface="org.osgi.service.http.HttpService"
/>
<!-- Bean declaration with properties for the RESTful service root context
     and OSGi HTTP service.  -->
<bean id="bean" class="com.oracle.cep.example.smartappliance.sink.AlertsService">
    <property name="rootContext" value="/alerts" />
    <property name="httpService" ref="httpServiceRef" />
</bean>
```

3. In your bean class code, provide a setter method for the HTTP Service, such as `setHttpService(HttpService httpService)`, as shown in Example 1–4.
4. In your bean class code, register the REST Servlet with the OSGi HTTP Service, as shown in Example 1–4.
    1. Create a Jersey `ServletContainer` instance with the appropriate initialization parameters.
    2. In the initialization parameters, set the `javax.ws.rs.Application` property as the implemented JAX-RS `Application` class name. The `Application` class is the one which loads the resource classes.

Example 1–4 Registering the OSGi HTTP Service in Class Code

```
public void setHttpService(HttpService httpService) throws Exception
{
    Hashtable<String, String> initParams = new Hashtable<String, String>();
    initParams.put("javax.ws.rs.Application", WebApplication.class.getName());
    Servlet jerseyServlet = new ServletContainer();
    httpService.registerServlet(rootContext, jerseyServlet, initParams, null);
}
```

1.5 Deploying Oracle Event Processing for Oracle Java Embedded Applications

You deploy Oracle Event Processing for Oracle Java Embedded applications by using the Deployer command-line tool included in the installation. You can deploy over both SSL and or non-SSL connections.

The Deployer tool included in Oracle Event Processing for Oracle Java Embedded is nearly identical to that provided in Oracle Event Processing. However, cluster and group deployments are not supported in Oracle Event Processing for Oracle Java Embedded.

For more about Deployer, see "Deployer Command-Line Reference" in *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing.*

Note: Oracle Event Processing Visualizer is not included with Oracle Event Processing for Oracle Java Embedded, so deployment through a graphical user interface is not supported.

To deploy an application to a running Oracle Event Processing for Oracle Java Embedded server, use a command such as the following:

```
java -jar wlevsdeploy.jar -url http://hostname:9002/wlevsdeployer -user wlevs
-password wlevs1 -install myoepapp.jar
```

The following examples illustrate additional deployment scenarios.

Deploying to a Local Server

```
java -jar wlevsdeploy.jar -user wlevs -password wlevs -url
http://localhost:9002/wlevsdeployer -install application
```

Deploying Over a Non-SSL Connection

```
java -jar wlevsdeploy.jar -user wlevs -password wlevs -url
http://<host>:9002/wlevsdeployer -install application
```

Deploying Over SSL

```
java -Djavax.net.ssl.trustStore=d:\downloads\evsidentity.jks
-Djavax.net.ssl.trsutStorePassword=welcome1 -jar wlevsdeploy.jar -user wlevs
-password wlevs -url https://<host>:9003/wlevsdeployer -install application
```

1.6 Administering Oracle Event Processing for Oracle Java Embedded Applications

You can administer Oracle Event Processing for Oracle Java Embedded applications using the wlevs.Admin command-line tool and Java Management Extensions (JMX).

Whether you use JMX or wlevs.Admin, Oracle Event Processing for Oracle Java Embedded supports administration for all but those features omitted from this release. For a list of omitted features, see Section 1.1.2, "Features Included in Oracle Event Processing for Oracle Java Embedded".

Note: Oracle Event Processing Visualizer is not included with Oracle Event Processing for Oracle Java Embedded, so administration through a graphical user interface is not supported.

Administration with JMX

Oracle Event Processing for Oracle Java Embedded supports remote JMX connections via MSA RMI as in Oracle Event Processing. Most configuration and runtime MBeans are supported, with the exception of those specifically related to omitted features. For example the HttpPubSubAdapterMBean, JMSAdapterMBean, and Monitoring related MBeans are not supported because the underlying features are not supported.

Note that the JES JRE does not include support for the JMX local attach API, so "local" (non-RMI) JMX connections aren't supported.

For more about using JMX, see "Configuring JMX for Oracle Event Processing" in *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.

Administration with wlevs.Admin

The Admin command line utility is supported and includes most of the functionality present in Oracle Event Processing. Unsupported are those options specifically related to missing features, including commands related to monitoring or record-playback.

For example, you can add an Oracle CQL rule locally with the Admin utility's ADDRULE command, as shown in the following example:

```
prompt> java wlevs.Admin
        -url service:jmx:msarmi://localhost:9002/jndi/jmxconnector
        -username wlevs -password wlevs
        ADDRULE -application helloworld -processor helloworldProcessor
        -query myquery "SELECT * FROM Withdrawal [Rows 5]"
```

For more about using the Admin tool, see "wlevs.Admin Command-Line Reference" in *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.

Note: This release does not support starting the server with the `-disablesecurity` flag.

1.6.1 The policygen Command-Line Utility

Use the policygen command-line utility to generate an extensible access control markup language (XACML) file.

The `policygen` utility is located at `ORACLE_OEP_HOME/ocep_11.1/bin/policygen.sh`, where *`ORACLE_OEP_HOME`* refers to the Oracle Event Processing for Oracle Java Embedded installation directory.

1.6.1.1 policygen Syntax

```
policygen.sh [-h] [-s] [-l|-x] entitlement_file xacml_output_file
```

where:

Table 1–6 policygen Arguments

Option	Description	Default Value
`-s`	Specifies that standard XACML policy will be generated in the output file; otherwise, entitlement XACML policy will be generated.	
`-l`	Specifies that an XACML LDIFT file should be generated.	
`-x`	Specifies that an XACML policy file should be generated.	
`-h`	Displays usage help.	
`entitlement_file`	Specifies the location of input entitlement XML file.	
`xacml_output_file`	Specifies the location of output XACML file.	

1.6.1.2 Examples of Using policygen

For example:

```
prompt> policygen.sh ~/security/policygen/AuthorizerInit.xml
XACMLAuthorizerInit.ldift
```

1.7 Supported Platform and Resource Configurations

Oracle Event Processing for Oracle Java Embedded supports the following platform configurations:

- **Java Runtime Environment (JRE):** Java Embedded Suite (JES) 7.0
- **Operating System:** Linux
- **Hardware:** x86 or ARM

Note that compared with Oracle Event Processing, Oracle Event Processing for Oracle Java Embedded is designed to support much smaller disk and heap footprint requirements.

For a new installation, Oracle Event Processing for Oracle Java Embedded requires 60 MB of disk space. With the server started and applications deployed, additional disk space is needed per application.

2 Installing Oracle Event Processing for Oracle Java Embedded

This chapter describes how to install Oracle Event Processing for Oracle Java Embedded, a server for developing and running event processing applications.

This chapter includes the following sections:

- Section 2.1, "Installation Overview"
- Section 2.2, "Installing Oracle Event Processing for Oracle Java Embedded from the Command Line"
- Section 2.3, "Post-Installation Steps"

2.1 Installation Overview

Oracle Event Processing for Oracle Java Embedded does not include an installation program. To install it, you extract the contents of its distribution file to the location where you want to install it.

Oracle Event Processing for Oracle Java Embedded is designed for use with the Java Embedded Suite (JES). The installation instructions provided here assume that your installation of Oracle Event Processing for Oracle Java Embedded will use the JRE provided with JES.

To install Oracle Event Processing for Oracle Java Embedded:

1. Download and install Java Embedded Suite (JES).
2. Download and install Oracle Event Processing for Oracle Java Embedded.
3. Set `JAVA_HOME` to use the JES JRE.

For more detailed step-by-step instructions, including download location, see Section 2.2, "Installing Oracle Event Processing for Oracle Java Embedded from the Command Line".

2.1.1 Before You Install

Oracle Event Processing for Oracle Java Embedded is designed to work with the Java Embedded Suite (JES), including the JRE bundled with JES. Because of this, JES is a prerequisite for using Oracle Event Processing for Oracle Java Embedded.

The installation instructions include notes about JES installation.

For additional system requirements, see Section 1.7, "Supported Platform and Resource Configurations".

2.1.2 Oracle Event Processing for Oracle Java Embedded Directory Structure and Concepts

The Oracle Event Processing for Oracle Java Embedded distribution file includes the event server, a default domain, and a sample application.

When you extract the Oracle Event Processing for Oracle Java Embedded distribution file, the directory structure includes the following noteworthy items:

Table 2–1 Oracle Event Processing for Oracle Java Embedded Directory Structure

Path	Description
`OEPEMBEDDED_HOME/modules`	Libraries comprising product features.
`OEPEMBEDDED_HOME/ocep_11.1/bin`	Configuration and deployment tools.
`OEPEMBEDDED_HOME/ocep_11.1/common`	Tools and examples for domain configuration.
`OEPEMBEDDED_HOME/ocep_11.1/examples`	Sample application code.
`OEPEMBEDDED_HOME/ocep_11.1/modules`	Libraries comprising product features.
`OEPEMBEDDED_HOME/ocep_11.1/utils`	Database and security utilities.
`OEPEMBEDDED_HOME/ocep_11.1/xsd`	XML schemas that define XML configuration files.
`OEPEMBEDDED_HOME/user_projects`	Default domain for user projects.

2.2 Installing Oracle Event Processing for Oracle Java Embedded from the Command Line

You install Oracle Event Processing for Oracle Java Embedded by extracting the contents of the installation .zip file to the directory of your choosing.

Note: There is no graphical installer in this release.

To install Oracle Event Processing for Oracle Java Embedded:

1. Log in to the device on which you want to install Oracle Event Processing for Oracle Java Embedded. Be sure you log in as the user that will be the main administrator of the installation.
2. Download the Oracle Event Processing for Oracle Java Embedded distribution ZIP file. For example, at the command line use the following command:

   ```
   wget http://location/of/ofm_oep_embedded_RELEASE_NUMBER_linux.zip
   ```

   For example:

   ```
   wget http://location/of/ofm_oep_embedded_11_1_1_7_0_linux.zip
   ```

3. Extract the file to the installation location. For example, use the following command:

   ```
   unzip ofm_oep_embedded_RELEASE_NUMBER_linux.zip
   ```

4. If you haven't already, browse the Oracle web site for the JES version you need. Browse at the following URL:

   ```
   http://www.oracle.com/technetwork/java/embedded/downloads/java-embedded-suite/i
   ndex.html
   ```

5. Download the Java Embedded Suite distribution ZIP file. For example, use a command such as the following:

```
wget
http://download.oracle.com/otn/java/JES/7.0-b11/jes-7.0-ga-bin-b11-linux-arm-ru
ntime-15_nov_2012.zip
```

6. Extract the file to the Java home location. For example, use a command such as the following:

```
unzip jes-7.0-ga-bin-b11-linux-arm-runtime-15_nov_2012.zip -d
home/user/java/jes7.0/jre
```

7. Ensure that you have a `$JAVA_HOME` system variable and set it to the path where you unzipped the embedded JRE.

```
export JAVA_HOME=/home/user/java/jes7.0/jre
```

2.3 Post-Installation Steps

For information about configuring and domain creation, see the following sections:

- Section 1.3, "Creating Domains and Configuring the Server"
- Section 1.2, "Oracle Event Processing for Oracle Java Embedded Application and Development Framework"