# Oracle® Fusion Middleware

Getting Started Guide for Oracle Event Processing

11*g* Release 1 (11.1.1.9)

**E14476-11**

April 2015

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

ORACLE

Oracle Fusion Middleware Getting Started Guide for Oracle Event Processing 11*g* Release 1 (11.1.1.9)

E14476-11



Primary Author: Oracle Corporation

# Contents

## List of Examples

## List of Figures

## List of Tables

# Preface

This document provides general background information and detailed code samples to help you learn about Oracle Event Processing and the Oracle Continuous Query Language (Oracle CQL).

## Audience

This document is intended for users interested in learning about Oracle Event Processing and Oracle CQL. 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 that have purchased support 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:

- *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 Visualizer User's Guide for Oracle Event Processing*
- *Oracle Fusion Middleware Java API Reference for Oracle Event Processing*
- *Oracle Fusion Middleware CQL Language Reference for Oracle Event Processing*
- *Oracle Database SQL Language Reference* at http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/toc.htm
- SQL99 Specifications (ISO/IEC 9075-1:1999, ISO/IEC 9075-2:1999, ISO/IEC 9075-3:1999, and ISO/IEC 9075-4:1999)
- Oracle Event Processing Forum: http://forums.oracle.com/forums/forum.jspa?forumID=820

- Oracle Event Processing Samples:
  http://www.oracle.com/technologies/soa/complex-event-processing.html

## 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** | Product renamed to Oracle Event Processing. | X |
| **Chapter 1 Overview of Oracle Event Processing** | | |
| Section , "Introduction to Oracle Event Processing" | Introduction now includes contextual information about uses of Oracle Event Processing applications and use cases. | X |
| Section , "Oracle Event Processing Application and Development Framework" | Section rewritten with more introductory information about key concepts, event processing networks, the application development process, and supporting technologies. | X |
| **Former Chapter 2 Oracle Event Processing Samples** | | |
| | Chapter moved to Oracle Event Processing Samples in *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*. | X |

# 1

# Overview of Oracle Event Processing

This chapter provides an overview of Oracle Event Processing. It describes key concepts, features, and use cases, including event processing networks, developing with the Eclipse IDE, and managing applications with Oracle Event Processing Visualizer.

This chapter includes the following sections:



## Introduction to Oracle Event Processing

Oracle Event Processing is middleware for creating applications that receive potentially large amounts of streaming data representing events, then respond in real time based on the event data. Oracle Event Processing applications can receive event data from sources near and remote, including from sensors (such as for environment conditions), social media, financial feeds, and moving sources such as vehicles or mobile devices.

Some of the logic in Oracle Event Processing applications is typically implemented as Oracle Continuous Query Language (Oracle CQL) queries that examine event data in real time as it is received, filtering the data or watching for patterns. Other logic implemented in Java code can respond to discoveries made from the data, such as by initiating other processes or sending related data to other components (including reporting tools for display in a user interface).

Through connections with external components and resources -- including relational databases, "big data" sources such as Hadoop and NoSQL, Java Message Service (JMS) systems, and caches -- Oracle Event Processing applications can combine or correlate event data with other kinds of data. In this way, these applications integrate ongoing streaming data with existing, relatively static systems.

### Very Large Amounts of Data from a Growing List of Sources

Oracle Event Processing provides a way to take advantage of the world's ever-growing amount of event data. Both on public networks such as the Internet and private, corporate networks, the increasing number of event sources and amount of

event data provides opportunities that event-driven applications can take advantage of.

Sources of events include:

- Applications on mobile devices such as smartphones can provide location data as events that include geographical coordinates.
- Social media such as Twitter can offer events related to tweets that mention specific companies or products.
- Devices such as the sensors in a data center can send regular, frequent events that contain information about environmental conditions in the rooms containing servers.

The events in each of these examples represent a potentially huge amount of streaming data. To take advantage of the data, an application should be able to filter out irrelevancies and identify patterns of interest that can be used to a company's advantage or to defuse immediate business threats. In many cases, making the most of events means being able to execute application logic in response to particular patterns -- and to do so in real time.

In this context, with so much data moving so quickly, an asynchronous processing model event is needed to ensure that processing scales well. Oracle Event Processing is designed to support very low latency and high throughput in an application context that assumes a large amount of data and a high requirement for immediate response.

## Oracle Event Processing High Level Use Cases

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

### Financial: Responsive Customer Relationship

Acting on an initiative to improve relationships with customers, a retail bank designs an effort to provide coupons tailored to each customer's purchase pattern and geography.

The bank collects automated teller machine (ATM) data, including about the geographical region for the customer's most common ATM activity. The bank also captures credit card transaction activity. Using this data, the bank can push purchase incentives (such as coupons) to the customer in real time based on where they are and what they tend to buy.

An Oracle Event Processing application receives event data in the form of ATM and credit card activity. Oracle CQL queries filter incoming events for patterns that isolate the customer's geography (via GPS coordinates) and likely purchase interests nearby. This transient data is matched against the bank's stored customer profile data. If a good match is found, a purchase incentive is sent to the customer in real time, such as through their mobile device.

### Telecommunications: Real-Time Billing

Due to significant growth in mobile data usage, a telecommunications company with a large mobile customer base wants to shift billing for data usage from a flat-rate system to a real-time per-use system.

The company tracks IP addresses allocated to mobile devices and correlates this with stored user account data. Additional data is collect from deep-packet inspection (DPI) devices (for finer detail about data plan usage) and IP servers, then inserted into a Hadoop-based big data warehouse.

An Oracle Event Processing application receives usage information as event data in real time. Through Oracle CQL queries, and by correlating transient usage data with stored customer account data, the application determines billing requirements.

#### Energy: Improving Efficiency Through Analysis of Big Data

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. One application, deployed in each of thousands of data centers, receives sensor data as event data. Through Oracle CQL queries against events representing the sensor data, this application analyzes local usage, filtering for fault and problem events and sending alerts when needed. The other application, deployed in multiple central management sites, receives event data from lower-tier the applications. This application aggregates and correlates data from across the system to identify consistency issues and produce data to be used in reports on patterns.

## Oracle Event Processing Application and Development Framework

Understanding the Oracle Event Processing framework includes looking at it from the perspective of its key concepts, development artifacts and process, and constituent technologies.

The following sections describe concepts that are important to understanding how applications work

- Section , "Key Concepts in Oracle Event Processing"
- Section , "Overview of Event Processing Networks"
- Section , "Development Process and Lifecycle for Oracle Event Processing Applications"
- Section , "Technologies in Oracle Event Processing"

### Key Concepts in Oracle Event Processing

The concepts described in this section represent ideas that are unique to the event-driven model on which Oracle Event Processing applications are based. If you are new to this model, getting to know these concepts should help provide a basis for understanding more specific and technical aspects of Oracle Event Processing.

#### event data

Event data is data received from (or sent to) a component that is external to the EPN. Event data is typically in a form that is understandable to the external component with which Oracle Event Processing is communicating. For example, the data coming from a sensing device will likely be in a form native to that device.

Event data is made up of tuples, or ordered lists of elements. For example, the following list of values could be a tuple representing a trade of 50 shares of Oracle stock at 30.86 a share:

{orcl,30.86,50}

You can think of event data tuples as similar to rows in a database. The data has a schema -- here, {symbol as string,price as double,volume as integer} -- that defines a consistent shape for the data.

### event type

An event type provides structure to represent event data inside an EPN -- it normalizes event data received by the application. The recommended way to create an event type is to create a JavaBean in which properties correspond to elements in event data tuples.

When event data arrives at an EPN, it is bound to an instance of the event type that was developed to represent it. The event type instance -- usually referred to simply as an "event" -- provides developers with a predictable, normalized way to access the event data from code in EPN stages.

### stage

Connected stages in an EPN provide a way to execute different kinds of code against events passing through the EPN. Kinds of stages can include an adapter, a processor, and a bean.

For example, an adapter could receive event data and bind it to an event type instance, then pass the event along to a processor. The processor's Continuous Query Language (CQL) code can query the events (as SQL code queries database rows), looking for particular patterns in the data as it flows through the EPN. Events that meet the pattern criteria could be passed along to a bean (written in Java), where the data could be used in a calculation with data retrieved from an external source. A further bean could use the calculation result to execute a process using an external component.

### event sources and sinks

Any component designed to receive or send events in an EPN (such as EPN stages) must have been implemented specifically to do so. Components that are able to receive events are known as event sinks, while components that send events are known as event sources. A single component is usually both.

The set of stage components included in Oracle Event Processing, such as adapters and the components on which CQL processors are based, already support the required functionality. Developers can add event sink and source support to beans, new adapters, and other code they write by implementing interfaces from the OEP API.

### streams and relations

The concepts of streams and relations are unique to Oracle Event Processing applications. As a result, they can seem more esoteric than other application ideas. However, they're important for describing changes in the condition of events as a result of queries.

A stream of events is in sequential order by time -- one after the other. Events are by definition time-based, so a stream is that sense the natural condition of events. It is how event data arrives at an Oracle Event Processing application.

A relation is, in a manner of speaking, a bucket of related events. A relation is the result produced by many CQL queries. So you can think of a relation as being similar to the results of a database query. Events in a relation are, well, related by some criteria, such as those used in the query. They aren't in sequential order.

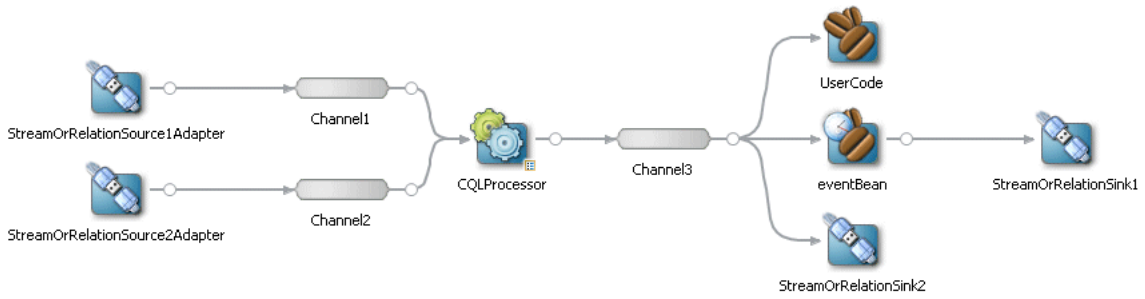
## Overview of Event Processing Networks

An event processing network (EPN) is the core of each Oracle Event Processing application. The application processes the events it receives as each event moves from one part of the application to another.

To process events with Oracle Event Processing, you build an application whose core is an EPN. The EPN is made up of stages that each serve a distinct role in processing events, from receiving event data to querying the data to executing logic based on what is discovered about the events. The application receives raw event data, binds the data to event types, then routes the events from stage to stage for processing.

Event processing networks improve the ability to integrate through separately developed components working correctly together. For example, one can add user code and reference to external services at several places in the network.

***Figure 1–1 Oracle Event Processing Application***




A developer creates an EPN by assembling stages in a sequence so that events can be filtered, analyzed, and responded in real time. Each stage in the sequence provides its own logic for filtering and responding to event data.

Oracle Event Processing includes support for the following kinds of stages:

- **Adapters** provide a way for the EPN to exchange data with components that are external to the network. Adapters understand the inbound and outbound protocols for the component they're interacting with. They are also responsible for converting the event data into the normalized form represented by an event type. Oracle Event Processing includes adapters for exchanging data as CSV files (for testing and debugging purposes), as well as with the Java Message Service (JMS) and an included HTTP Publish-Subscribe server. Developers can also create their own adapters using the Oracle Event Processing API.
- **Processors** contain query code written in Continuous Query Language (CQL) and consume normalized event data from a channel that precedes them in the network. As part of their work, processors may also generate new events to an output channel. CQL extends the database query language, SQL. CQL includes functionality specifically designed for querying streaming data, and is particularly useful for fine-grained queries of events. Within a processor, a CQL developer can write multiple query blocks in which one block executes against the results of another.
- **Bean** stages provide a place for logic written in Java. Java code in a bean implements Oracle Event Processing API interfaces through which the code can receive events (an event sink), send events (an event source), or both. A developer can implement a bean following either the standard Spring bean model or the OEP event bean model, which supports OEP-specific functionality. Support for the Spring bean model is particularly useful in cases where existing Spring bean code can be leveraged in the EPN.

- **Channels** connect some kinds of stages, and also feature configuration options that can be used to tune and enhance performance. For example, channels can queue event data until a processor agent can act upon it.

The following very brief EPN topology examples suggest how stages can be combined to achieve certain goals:

- Adapter -> Channel -> Bean

  Scenario: Events do not need to be processed by an Oracle CQL query. Events received are only adapted from a proprietary protocol to some normalized model.

- Adapter -> Channel -> Processor -> Channel -> Bean

  Scenario: Events must be processed by a single Oracle CQL stage, then sent through to logic coded in Java. For example, the processor might filter to only those events that are relevant to action taken by logic in the bean.

- Adapter -> Channel -> Processor -> Channel -> Bean -> Channel -> Processor -> Channel -> Bean

  Scenario: Events pass through two layers of event processing, each with its own Oracle CQL query code. The first processor discovers causal relationships between events, while the second processor aggregates events to pass along those that are notable for logic implemented in the bean.

## Development Process and Lifecycle for Oracle Event Processing Applications

Developing event-driven applications with Oracle Event Processing is in most ways like developing applications on more common models, such as models for JEE applications. However, specific aspects of event-driven applications, in particular the time-oriented quality of working with streaming data, drive a need for specific tools and processes.

The following sections introduce the application development lifecycle as it pertains specifically to Oracle Event Processing applications.

The development stages described in this section assume that you have installed and configured the Oracle Event Processing server. This step installs the server, but also supporting functionality such as clustering, security, network I/O, Java Management Extensions (JMX), JDBC, HTTP Publish-Subscribe server, and logging.

Once you install and configure the server, the following stages include both development and administrative work.

1. **Design and develop** applications using an integrated development environment (IDE).
2. **Deploy** applications using an included command-line or browser-based tool.
3. **Configure** applications either by directly editing configuration artifacts or through an included browser-based tool.
4. **Test** applications with event data, including by injecting and tracing data against specific stages in an event processing network.
5. **Monitor** aspects of the application, including channels, cluster nodes, and caches.

The following sections provides more detailed information on each of these lifecycle phases.

### Designing and Developing

With application goals in hand, developers define the schema for events that the application will handle. Event schemas used within the application are based on event data received from external components (such as a monitoring device). The event properties in a schema is often a subset of the data received since only certain elements might be needed. Application designers also define for external components (if any) that the application will interact with, such as a cache containing identifying data needed in combination with event data to interact with an external component.

In developing the application, a developer builds out an event processing network by selecting and configuring stages in the network. The developer might write logic code for certain stages, such as CQL code in processors and Java code in beans.

For example, a CQL developer might add query code to a processor stage designed to identify events that are part of a particular pattern (such as a dramatic temperature increase within 10 minutes' time). Then, within a JavaBean added as an event bean stage, a developer could include logic in Java to take action based on the content of events found in the pattern (for example, by sending an alert using an event property value that indicates where the temperature increase is occurring).

All of this is most easily done using an integrated development environment (IDE) that includes design tools specifically created for building Oracle Event Processing applications. Current releases of Oracle Event Processing include an Eclipse plugin with these tools. With an IDE, a developer can iteratively code, deploy, and debug an application until it is ready for testing and production.

### Deploying

Oracle Event Processing applications support deployment to standalone and multi-server domains. You can use one of two methods to deploy an application for testing and production: the Deployer command-line utility and Oracle Event Processing Visualizer. (During development, it's common to use an IDE to deploy an application in progress.)

The Deployer is a command-line utility that uses HTTP to connect to the OEP server and deploy the specified application. Deployer can also be used for other operations, such as suspending, resuming, and uninstalling.

Oracle Event Processing Visualizer is a browser-based tool for managing OEP servers and applications. The tool simplifies deployment and runtime configuration for both servers and the applications that run on them.

### Configuring

Oracle Event Processing provides multiple layers of configuration for tuning. In addition to full configuration options for the server itself, Oracle Event Processing provides configuration for event processing network stages and other application characteristics. In particular, Oracle CQL queries are configurable at runtime.

Many of these configuration options may be used to tune the server and applications for performance and high availability.

### Testing

In addition to providing server and application management, Oracle Event Processing Visualizer provides a means to test an application by tracing and injecting events. Event tracing displays messages for events as they pass through the EPN. With event injection, you can inject an event into a particular EPN stage in order to discover or confirm how that stage handles an event with particular properties.

### Monitoring

Through the Oracle Event Processing Visualizer and programmatic resources, you can monitor aspects of an application in production. For example, Oracle Event Processing Visualizer provides the ability to monitor event throughput and latency. It also provides the ability to monitor Coherence cluster nodes and caching.

## Technologies in Oracle Event Processing

The Oracle Event Processing platform is made up of standard technologies that provide functionality for developing application logic, as well as deploying and configuring applications.

- Java

  Much of the Oracle Event Processing server's functionality is written in the Java programming language. Java is the language you use to write some components that make up an OEP application. These include event beans and Spring beans.

- Spring

  Spring is a collection of technologies designed to ease and augment enterprise application development with Java. Oracle Event Processing makes significant use of the Spring configuration model, which developers use to connect and configure parts of an application. OEP applications also support adding logic as Spring beans, Java components that support Spring framework features.

  You can find out more about Spring at the project's web site:
  http://www.springsource.org/get-started

- OSGi

  OSGi (formerly the Open Services Gateway initiative) is the module system through which Oracle Event Processing application components are assembled in a deployment. By following OSGi component guidelines, OEP application components support a modular assembly model that eases management.

  You can find out more about OSGi at wikipedia:
  http://en.wikipedia.org/wiki/OSGi

- XML

  As with many other enterprise applications, Oracle Event Processing application configuration files are written in XML. These files include the EPN assembly file, which defines relationships between EPN stages, along with other design-time configuration. A separate configuration XML file contains configuration that can be modified after the application is deployed, including Oracle CQL queries.

- SQL

  Structured Query Language (SQL) is the basis for Oracle Continuous Query Language (Oracle CQL), which extends SQL with functionality designed to address the needs of applications that use streaming data.

- Hadoop

  Oracle CQL developers can access "big data" Hadoop data sources from query code. Hadoop is an open source technology that provides access to large data sets that are distributed across clusters. One strength of the Hadoop software is that it provides access to large quantities of data not stored in a relational database.

  If you need more information about Hadoop, you might start with the Hadoop project web site at `http://hadoop.apache.org/`.

- NoSQL

  Oracle CQL developers can access "big data" NoSQL data sources from query code. The Oracle NoSQL Database is a distributed key-value database. In it, data is stored as key-value pairs, which are written to particular storage node(s). Storage nodes are replicated to ensure high availability, rapid failover in the event of a node failure and optimal load balancing of queries.

  If you need more information about Oracle NoSQL, be sure to see its Oracle Technology Network page at http://www.oracle.com/technetwork/products/nosqldb/.

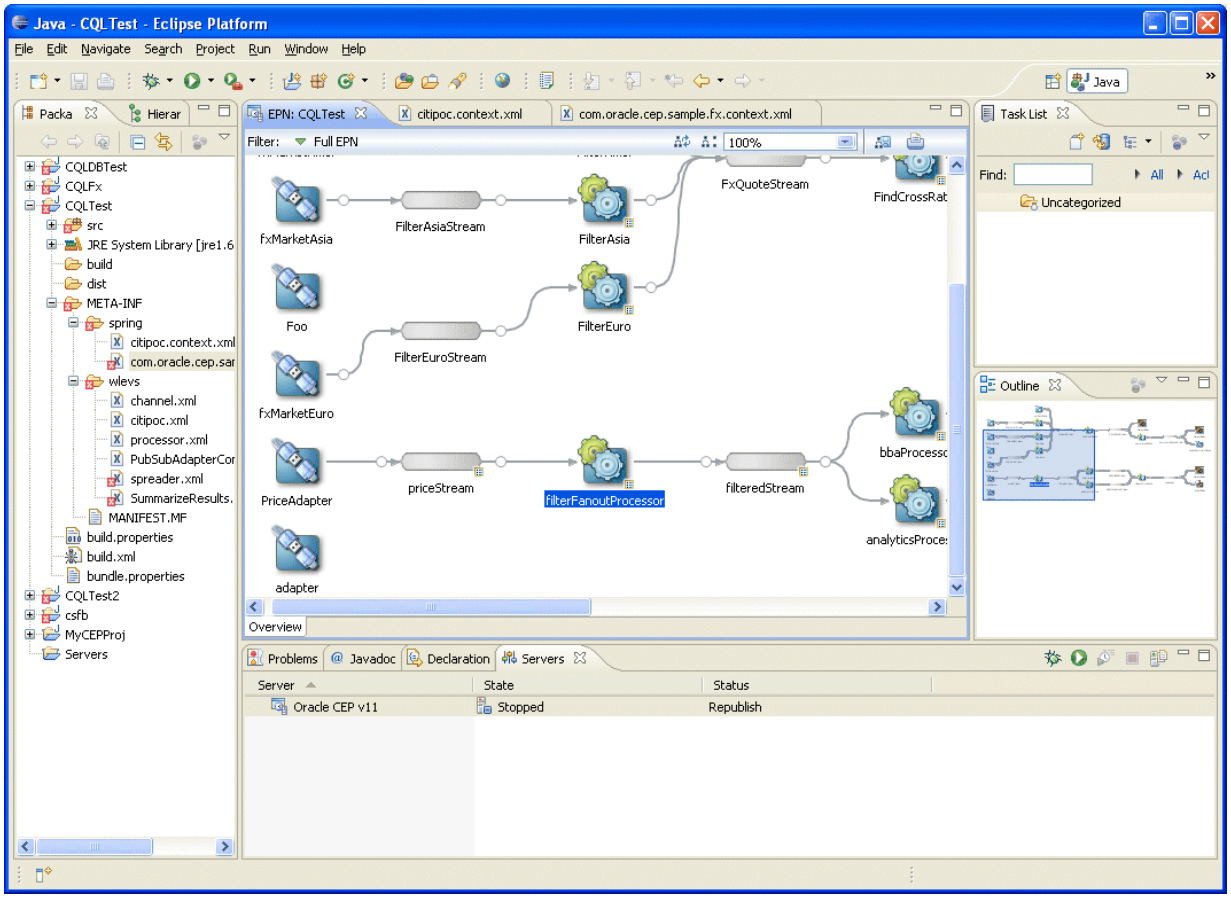
# Development and Administrative Tools

Oracle Event Processing includes tools designed to ease development and management of the Oracle Event Processing server and applications.

### Oracle Event Processing IDE for Eclipse

Oracle Event Processing includes a plugin for use with the Eclipse integrated development environment (IDE). The Oracle Event Processing IDE for Eclipse includes features specifically designed to ease development of Oracle Event Processing applications. Figure 1–2 shows the IDE open to show the editor for designing an event processing network (EPN).

***Figure 1–2 Oracle Event Processing IDE for Eclipse***



The Oracle Event Processing IDE for Eclipse is a set of plugins for the Eclipse IDE designed to help develop, deploy, and debug Oracle Event Processing applications.

The key features of Oracle Event Processing IDE for Eclipse are:

- Project creation wizards and templates to quickly get started building event driven applications.
- Advanced editors for source files including Java and XML files common to Oracle Event Processing applications.
- Integrated server management to seamlessly start, stop, and deploy to Oracle Event Processing server instances all from within the IDE.
- Integrated debugging.
- Event processing network (EPN) visual design views for orienting and navigating in event processing applications.
- Integrated support for the Oracle Event Processing Visualizer so you can use the Oracle Event Processing Visualizer from within the IDE (see Oracle Event Processing Visualizer).

For more information, see:

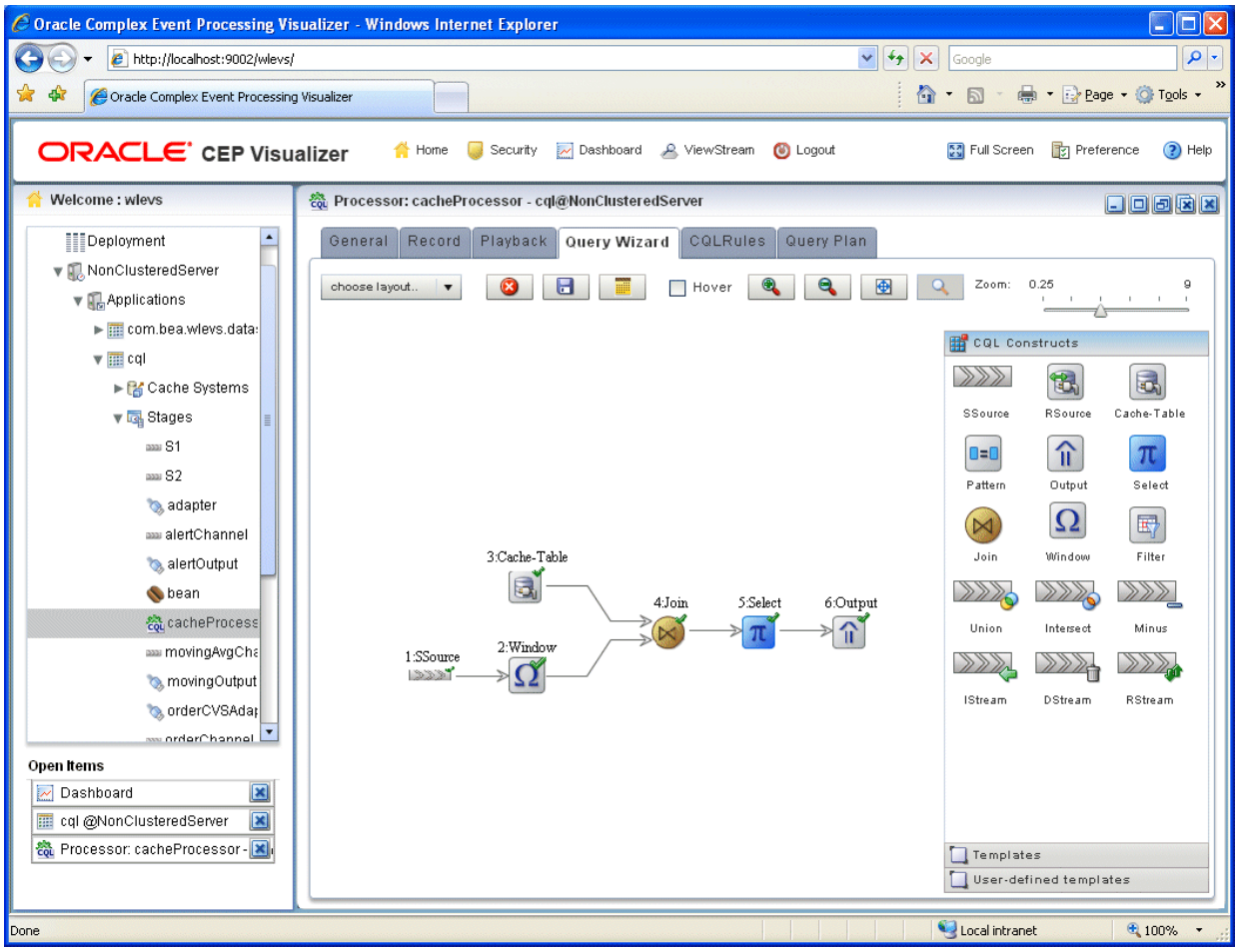
- Section , "Installing the Oracle Event Processing IDE for Eclipse"
- *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*

### Oracle Event Processing Visualizer

Oracle Event Processing provides an administration console called the Oracle Event Processing Visualizer. This tool provides access to server and application configuration options that are available at runtime, including the ability to edit Oracle CQL queries and to test an application by injecting events.

Figure 1–3 shows Oracle Event Processing Visualizer displaying the Oracle CQL design user interface.

***Figure 1–3 Oracle Event Processing Visualizer***




Using Oracle Event Processing Visualizer, you can manage, tune, and monitor Oracle Event Processing server domains and the Oracle Event Processing applications you deploy to them all from a browser. Oracle Event Processing Visualizer provides a variety of sophisticated run-time administration tools, including support for Oracle CQL and EPL rule maintenance and creation.

Oracle Event Processing Visualizer is pre-installed in every Oracle Event Processing server.

For more information, see *Oracle Fusion Middleware Visualizer User's Guide for Oracle Event Processing*

## Documentation Included in This Release

Oracle Event Processing provides the following documentation.

*Oracle Fusion Middleware Getting Started Guide for Oracle Event Processing*

Use this guide to get an introduction to Oracle Event Processing. The guide describes where it is useful and what its framework is made up of.

*Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*

Use this guide to deploy and manage Oracle Event Processing applications. It describes how to manage Oracle Event Processing in single- and multi-server

domains. It also describes how to configure component parts of an installation, including security, Jetty, JMX, and JDBC.

*Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*

This guide describes how to develop Oracle Event Processing applications. It introduces key concepts at a more technical level. It also describes the parts of a typical application, along with how to assemble them.

*Oracle Fusion Middleware Visualizer User's Guide for Oracle Event Processing*

Use this guide when using the browser-based Oracle Event Processing Visualizer to manage and configure the Oracle Event Processing server and applications deployed on it.

*Oracle Fusion Middleware CQL Language Reference for Oracle Event Processing*

Use this reference for descriptions and syntax of the Oracle CQL language. Oracle CQL extends Structured Query Language (SQL) to address the particular needs of querying streaming data.

*Oracle Fusion Middleware EPL Language Reference for Oracle Event Processing*

This reference describes the deprecated Oracle Event Processing Language (EPL). All new development should use Oracle Continuous Query Language instead. Support and documentation for this language will be removed in a future release.

## Supported Platforms

For detailed information on the platforms that Oracle Event Processing supports, see: http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html.

You can find the installation program appropriate for your platform here:

http://www.oracle.com/technetwork/middleware/complex-event-processing/downloads/index.html. For more information, see Section , "Installation Overview".

## Next Steps

- Install Oracle Event Processing 11g Release 1 (11.1.1).

  See Chapter 2, "Installing Oracle Event Processing."

- Understand how the sample applications have been programmed by viewing the source and configuration files and then building them from their respective source directories.

  For more about the samples, see "Oracle Event Processing Samples" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

- Create your own Oracle Event Processing domain.

  See:

  - "Creating an Oracle Event Processing Standalone-Server Domain" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*
  - "Creating an Oracle Event Processing Multi-Server Domain Using Oracle Coherence" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*

  - "Creating an Oracle Event Processing Multi-Server Domain Using Oracle Event Processing Native Clustering" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*

- Create a new Oracle Event Processing application and deploy it to your new domain.

  See "Overview of Creating Oracle Event Processing Applications" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

- Pose questions and find solutions using the Oracle Event Processing forum: http://forums.oracle.com/forums/forum.jspa?forumID=820

# 2

# Installing Oracle Event Processing

This chapter describes how to install and upgrade Oracle Event Processing, including development tools for use with the Eclipse IDE.

This chapter includes the following sections:



## Installation Overview

To install Oracle Event Processing 11*g* Release 1 (11.1.1.9):

1. Download the Oracle Event Processing installer appropriate for your platform.

   See Section , "Supported Platforms".

2. Choose the JVM you will use.

   See Section , "Choosing Your JVM"

3. Decide if you are installing in a production environment or development environment.

   See Section , "Default Oracle Event Processing Domain ocep_domain and Samples".

4. Familiarize yourself with Oracle Fusion Middleware directory structure and concepts.

   See Section , "Oracle Fusion Middleware Directory Structure and Concepts".

5. Install Oracle Event Processing using the installer mode of your choice.

   See Section , "Installation Mode".

6. Install Oracle Event Processing patches, if required.

   See Section , "Installing an Oracle Event Processing Patch".

7. Perform post-installation tasks, if applicable.

   See Section , "Post-Installation Steps".

8. Decide whether or not you need to upgrade Oracle Event Processing and Oracle Event Processing applications to the current release.

   See Section , "Upgrading to Oracle Event Processing 11g Release 1 (11.1.1.9)".

9. Install Apache Ant, a Java-based build tool.

   See the Apache Ant Project at `http://ant.apache.org/`.

## Before You Start the Installation Program

Before you start the installation program, review the following information:

- If you are using the Generic Package installer or the Linux 64-bit installer to install Oracle Event Processing on a UNIX or Linux system, Oracle recommends that you set the `umask` to `027` on your system prior to installation. This ensures that WebLogic Server file permissions will be set properly during installation. Use the following command:

  ```
  umask 027
  ```

  You must enter this command in the same terminal window from which you plan to run the Oracle Event Processing installer.

- You must run the installer on a system whose locale is set to one using the English language.

## Choosing Your JVM

Oracle Event Processing supports the following Java Virtual Machines (JVM):

- Section , "Oracle JRockit Real Time"
- Section , "Sun JVM"
- Section , "Other Platform-Specific JVMs"

### Oracle JRockit Real Time

By default, Oracle Event Processing includes its own version of JRockit (in *MIDDLEWARE_HOME*/jrockit_*JAVA-VERSION*_R*JROCKIT-VERSION*), but it does not include the deterministic garbage collector.

> **Note:** The Oracle Event Processing installer for the Sun JVM does not include a version of JRockit.

Oracle Event Processing performs optimally when it can access certain features from Oracle JRockit Real Time, in particular the JRockit deterministic garbage collector.

If your application requires low latency, optionally install Oracle JRockit Real Time.

> **Caution:** Be sure you install the version of Oracle JRockit Real Time for Java version 5.0 or 6.0. Oracle JRockit Real Time for Java version 1.4.2 is not compatible with this version of Oracle Event Processing.

For more information on Oracle JRockit Real Time, see http://www.oracle.com/technology/products/jrockit/jrrt/index.html.

### Sun JVM

The Sun JVM is the default JVM on Solaris SPARC and is included in the Solaris SPARC installer. If you want to use the Sun JVM (instead of JRockit) on platforms other than Solaris, download the JVM from the Oracle Java SE download site.

For more information, see:

- Section , "Supported Platforms"
- http://www.oracle.com/technetwork/java/javase/downloads/index.html

### Other Platform-Specific JVMs

For other platforms (such as HP or IBM/AIX), ensure that you have the appropriate platform-specific JVM installed.

For more information, see:

- Section , "Supported Platforms"
- Section , "Configuring Oracle Event Processing for the IBM JDK"

## Default Oracle Event Processing Domain ocep_domain and Samples

When you choose a **Typical** install, the installation does not include the default `ocep_domain` domain (with default passwords) and the product samples.

If you want to install the default `ocep_domain` and samples (recommended), choose the **Custom** option.

The **Typical** install is appropriate for a production environment while the **Custom** install is appropriate for a development environment.

## Oracle Fusion Middleware Directory Structure and Concepts

When you install Oracle Event Processing, consider the following scenarios:

- Section , "Oracle Event Processing-Specific Middleware Home"
- Section , "Existing Oracle Fusion Middleware Home"

For more information, see "Understanding Oracle Fusion Middleware Concepts and Directory Structure" in the *Oracle Fusion Middleware Installation Planning Guide*.

### Oracle Event Processing-Specific Middleware Home

In this scenario, you install Oracle Event Processing in a stand-alone configuration, in its own Middleware Home directory as Figure 2–1 shows.

*Figure 2–1 Oracle Event Processing-Specific Middleware Home*

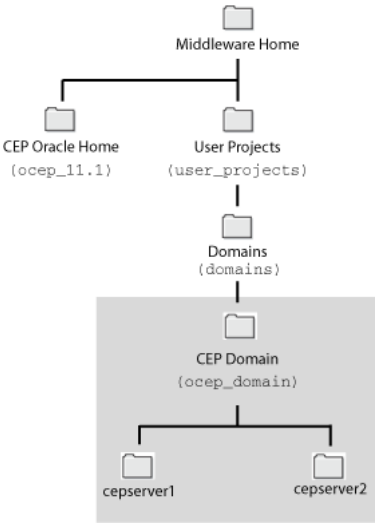



Table 2–1 lists the various home directories applicable to Oracle Event Processing in this scenario.

*Table 2–1 Home Directories and Oracle Event Processing-Specific Middleware Home*

| Home Directory Type | Home Directory |
|---|---|
| Middleware Home | As selected by the user at Oracle Event Processing install time. |
| WebLogic Server Home | N/A |
| Oracle Product Home[1] | `ocep_11.1` |
| Oracle Common Home | N/A |
| Domain | *`MIDDLEWARE_HOME`*`/user_projects/domains/ocep_domain` |

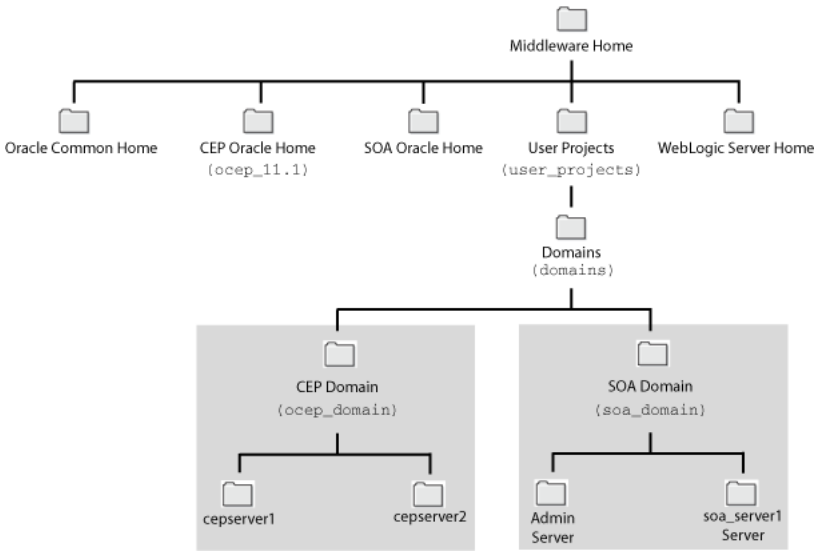
[1] Also known as simply the Oracle Home.

## Existing Oracle Fusion Middleware Home

In this scenario, you install Oracle Event Processing into an existing Oracle Fusion Middleware home as Figure 2–2 shows.

In this scenario, you install Oracle Event Processing into its own Product Oracle Home but you create Oracle Event Processing domains in the existing *`ORACLE_FUSION_MIDDLEWARE_HOME`*`/user_projects/domains` directory as Figure 2–2 shows.

*Figure 2–2 Oracle Event Processing in an Existing Oracle Fusion Middleware Home*




In this example topology, there are two Oracle products installed in the same Middleware Home:

- Oracle Event Processing
- Oracle SOA Suite

Table 2–2 lists the various home directories applicable to Oracle Event Processing in this scenario.

*Table 2–2 Home Directories and Existing Middleware Home*

| Home Directory Type | Home Directory |
|---|---|
| Middleware Home | As selected by the user at Oracle Fusion Middleware install time. |
| WebLogic Server Home | As determined by Oracle Fusion Middleware installer. |
| Oracle Product Home[1] | `ocep_11.1` |
| Oracle Common Home | As determined by Oracle Fusion Middleware installer. |
| Domain | *`MIDDLEWARE_HOME`*`/user_projects/domains/ocep_domain` |

[1] Also known as simply the Oracle Home.

## Installation Mode

You use the Oracle Event Processing installer in the following modes:

- Section , "Graphical Mode"
- Section , "Console Mode"
- Section , "Silent Mode"

### Graphical Mode

Graphical-mode installation is an interactive, GUI-based method for installing your software. It can be run on both Windows and UNIX systems. See Section , "Installing Oracle Event Processing in Graphical Mode."

**Caution:** If you want to run graphical-mode installation, the console attached to the machine on which you are installing the software must support a Java-based GUI. All consoles for Windows systems support Java-based GUIs, but not all consoles for UNIX systems do. If you attempt to start the installation program in graphical mode on a system that cannot support a graphical display, the installation program automatically starts console-mode installation.

### Console Mode

Console-mode installation is an interactive, text-based method for installing your software from the command line, on either a UNIX system or a Windows system. See Section , "Installing Oracle Event Processing in Console Mode."

### Silent Mode

Silent-mode installation is a non-interactive method of installing your software that requires the use of an XML properties file for selecting installation options. You can run silent-mode installation in either of two ways: as part of a script or from the command line. Silent-mode installation is a way of setting installation configurations only once and then using those configurations to duplicate the installation on many machines. See Section , "Installing Oracle Event Processing in Silent Mode."

## Installing Oracle Event Processing in Graphical Mode

This section describes how to install using the Oracle Event Processing installer in graphical mode.

For more information, see Section , "Installation Overview".

**To install Oracle Event Processing in graphical mode:**

1. Log in to the Windows or UNIX computer on which you want to install Oracle Event Processing.

   Be sure you log in to the computer as the user that will be the main administrator of the Oracle Event Processing installation.

2. Download the product distribution file for the platform on which you want to install Oracle Event Processing.

3. Launch the installation program in graphical mode using the commands listed in the following table appropriate for your platform.

| Platform | Instructions |
|---|---|
| Windows | Using Windows Explorer, double-click the appropriate installation program file from its download directory. |

| Platform | Instructions |
|---|---|
| Linux or Solaris | Open a command window, change to the download directory, and enter these commands:<br>`prompt> chmod a+x filename`<br>`prompt> ./filename`<br>In the preceding commands, *filename* is the name of the installation program specific to your platform (for more information, see http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.html.)<br>If you want to create an installation log, use the `-log=full_path_to_log_file` option; for example:<br>`prompt> ./filename -log=C:\logs\server_install.log` |

After the installation program has finished loading, you will see the standard Welcome window.

4. Click **Next**.
5. In the Choose Middleware Home Directory window, you can specify either an existing Oracle Middleware Home directory or create a new one:
   a. To install into an existing Oracle Middleware Home directory:
      – Select **Use an Existing Middleware Home**.
      – Select an existing Oracle Middleware Home directory from the list on the right.
   b. To install into a new Oracle Middleware Home directory:
      – Select **Create a New Middleware Home**.
      – Click the **Browse** button to browse your computer to select an existing directory or click **Reset** to reset the directory to the default, C:\Oracle\Middleware.
      – Click **Open**.

   The Oracle Middleware Home directory is the main installation directory for Oracle Event Processing, such as `c:\oracle_cep`. You can have one or many Oracle Middleware Home directories on your computer, whichever suits your development and production environments best.

   For more information, see Section , "Oracle Fusion Middleware Directory Structure and Concepts".
6. Click **Next**.
7. In the Choose Install Type window, you can choose one of the following options:
   a. To install all Oracle Event Processing components *except* the samples and use the Oracle JRockit JVM included with Oracle Event Processing:
      * Select **Typical**.
      * Click **Next**.
      * Proceed to step 10.

      The installer program checks to see if the Oracle Middleware Home directory contains the version of JRockit required by this release of Oracle Event Processing:

- If it finds the required JRockit installation, it does not install a new one.
- If it does not find an appropriate JRockit installation, then the installer installs its own version in the Oracle Middleware Home directory.

**b.** To install all Oracle Event Processing components *including* the default `ocep_domain` domain (with default passwords) and the samples and select a previously installed Sun or platform-specific JVM (or use the Oracle JRockit JVM included with Oracle Event Processing):

* Select **Custom**.
* In the Choose Products and Components window, check the components you want to install, such as the product samples.
* Click **Next**.

---

**Note:** By default, the complete installation does not include the default `ocep_domain` domain (with default passwords) and the product samples. If you want to install the samples (recommended), choose the **Custom** option.

If you want to use a Sun or platform-specific JVM, you must choose the **Custom** option.

---

The installer program allows you to choose the JDK to use and to decide whether or not to install the Oracle JRockit JVM included with Oracle Event Processing.

**8.** In the JDK Selection window, you can choose the JDK for the Oracle Event Processing server.

Use the **Browse** button to select the Sun or platform-specific JDK you installed previously.

If you do not want the installer to install the Oracle JRockit JVM included with Oracle Event Processing, uncheck this item.

**9.** Click **Next**.

**10.** In the Choose Product Installation Directories window, you can change the default name of the Oracle Product Home directory for Oracle Event Processing, `ocep_11.1`.

Although you can name this directory anything you want, Oracle recommends that you use the default name for clarity and standardization. For example, the documentation assumes that the Oracle Product Home directory is `ocep_11.1`.

For more information, see Section , "Oracle Fusion Middleware Directory Structure and Concepts".

**11.** Click **Next**.

**12.** If you are installing on Windows, and you logged in as a user with Administrator privileges, then you will see the Choose Shortcut Location window where you can choose where you want the Start Menu folder to appear. The following table describes the options available:

| If you select . . . | The following occurs . . . |
|---|---|
| All Users | Recommended. All users registered on the machine are provided with access to the installed software. Subsequently, if users without Administrator privileges use the Configuration Wizard from this installation to create domains, Start menu shortcuts to the domains are not created. In this case, users can manually create shortcuts in their local Start menu folders, if desired. |
| Local user | Other users registered on this machine will not have access to the Start menu entries for this installation. |

If you logged in as a user without Administrator privileges, the Start menu entries are created in your user's local Start menu folder.

13. Click **Next**.
14. The Installation Summary window shows the products and components you are about to install, along with the approximate size in MB. This window is for your information only; to change the components to be installed, use the **Previous** button to return to the appropriate window.
15. Click **Next**.

    The installer program installs Oracle Event Processing. The Installation Complete window indicates that the product was installed successfully.
16. Click **Done** to exit the program.
17. Review the post-installation steps that Section , "Post-Installation Steps" describes.

## Installing Oracle Event Processing in Console Mode

This section describes how to install using the Oracle Event Processing installer in console mode.

Console-mode installation is an interactive, text-based method for installing your software from the command line, on either a UNIX or Windows system.

When installing in console-mode, respond to the prompts in each section by entering the number associated with your choice or by pressing Enter to accept the default. To exit the installation process, enter `exit` (or `x`, for short) in response to any prompt. To review or change your selection, enter `previous` (or `p`, for short) at the prompt. To proceed to the following window, enter `next` (or `n`, for short).

> **Note:** In the following procedure, Windows conventions (such as back-slashes in pathnames) are used, for example, `C:\oracle_cep\ocep_11.1`. When entering pathnames on a UNIX system, be sure to use UNIX conventions, instead. For example, use forward slashes in pathnames, such as `/oracle_cep/ocep_11.1`.

For more information, see Section , "Installation Overview".

### To install Oracle Event Processing in console mode:

1. Log in to the Windows or Linux computer on which you want to install Oracle Event Processing.

   Be sure you log in to the computer as the user that will be the main administrator of the Oracle Event Processing installation.

2. Download the product distribution file for the platform on which you want to install Oracle Event Processing.
3. Launch the installation program in console mode using the commands listed in the following table appropriate for your platform.

| Platform | Instructions |
|---|---|
| Windows | Open a command window, change to the download directory, and enter the following command:<br>`prompt> filename -mode=console`<br>In the preceding command, *filename* is the name of the installation program specific to your platform (for more information, see http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.html).<br>If you want to create an installation log, use the `-log=full_path_to_log_file` option; for example:<br>`prompt> filename -mode=console -log=C:\logs\server_install.log` |
| Linux | Open a command window, change to the download directory, and enter these commands:<br>`prompt> chmod a+x filename`<br>`prompt> ./filename -mode=console`<br>In the preceding commands, *filename* is the name of the installation program specific to your platform (for more information, see http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.html).<br>If you want to create an installation log, use the `-log=full_path_to_log_file` option; for example:<br>`prompt> ./filename -mode=console -log=C:\logs\server_install.log` |

4. At the Welcome prompt, type `next` (or `n` for short) or press **Enter** to continue with the installation process.
5. In the Choose Middleware Home Directory window, you can specify either an existing Oracle Middleware Home directory or create a new one:
   a. To install into an existing Oracle Middleware Home directory:
      – Type the number of the existing Oracle Middleware Home directory.
   b. To install into a new Oracle Middleware Home directory:
      – Type `1` to create a new Oracle Middleware Home directory.
      – The installation program guides you through the required steps to create the new Oracle Middleware Home.

        Be sure to enter the full path of the Oracle Middleware Home directory, for example `C:\oracle_cep2`.

        **Note:** Do not terminate the path with a file separator character. That is, enter `C:\mydir` and not `C:\mydir\`.

        If you specify a directory that does not exist, the installation program creates it for you.

   The Oracle Middleware Home directory is the main installation directory for Oracle Event Processing, such as `c:\oracle_cep`. You can have one or many

Oracle Middleware Home directories on your computer, whichever suits your development and production environments best.

For more information, see Section , "Oracle Fusion Middleware Directory Structure and Concepts".

6. Confirm your choice for Oracle Middleware Home directory and enter `next` (or `n`).
7. In the Choose Install Type window, you can choose one of the following options:
   a. To install all Oracle Event Processing components *except* the samples and use the Oracle JRockit JVM included with Oracle Event Processing:
      * Type `1` to choose a **Typical** install.
      * Proceed to step 9.

      The installer program checks to see if the Oracle Middleware Home directory contains the version of JRockit required by this release of Oracle Event Processing:
      – If it finds the required JRockit installation, it does not install a new one.
      – If it does not find an appropriate JRockit installation, then the installer installs its own version in the Oracle Middleware Home directory.
   b. To install all Oracle Event Processing components *including* the default `ocep_domain` domain (with default passwords) and the samples and select a previously installed Sun or platform-specific JVM (or use the Oracle JRockit JVM included with Oracle Event Processing):
      * Type `2` to choose a **Custom** install.
      * In the Choose Components to Install window, enter the numbers in brackets to toggle the components you want to install, such as the samples. To toggle a selection in the list, types its number. When a check mark appears next to the option, the option is selected. To unselect the option, enter its number again to remove the check mark
      * Enter `next` (or `n`) when you have chosen the components.

      ---

      **Note:** By default, the complete installation does not include the default `ocep_domain` domain (with default passwords) and the product samples. If you want to install the samples (recommended), choose the **Custom** option.

      If you want to use a Sun or platform-specific JVM, you must choose the **Custom** option.

      ---

      The installer program allows you to choose the JDK to use and to decide whether or not to install the Oracle JRockit JVM included with Oracle Event Processing.
8. In the JDK Selection window, you can choose the JDK for the Oracle Event Processing server:
   - To add a local JDK, select the Add Local JDK option (`1`).

     The installation program guides you through the required steps to add a local JDK.

     Be sure to enter the full path to the JDK directory, for example:

     ```
     C:\Program Files\Java\jdk1.6.0_14
     ```

To add additional JDKs, select `1` again.

- In the JDK Selection window, enter the numbers in brackets to toggle the JDKs you want. To toggle a selection in the list, types its number. When a check mark appears next to the option, the option is selected. To unselect the option, enter its number again to remove the check mark.

  If you do not want the installer to install the Oracle JRockit JVM included with Oracle Event Processing, uncheck this item.

- Enter `next` (or `n`) when you have selected the local JDK.

9. In the Choose Product Installation Directories window, you can change the default name of the Oracle Product Home directory for Oracle Event Processing, `ocep_11.1`.

   Although you can name this directory anything you want, Oracle recommends that you use the default name for clarity and standardization. For example, the documentation assumes that the Oracle Product Home directory is `ocep_11.1`.

   For more information, see Section , "Oracle Fusion Middleware Directory Structure and Concepts".

   Enter `next` (or `n`) when you are done.

10. If you are installing on Windows, and you logged in as a user with Administrator privileges, then you will see the Choose Shortcut Location window where you can choose where you want the Start Menu folder to appear. The following table describes the options available:

| If you select . . . | The following occurs . . . |
|---|---|
| 1 "All Users" | Recommended. All users registered on the machine are provided with access to the installed software. Subsequently, if users without Administrator privileges use the Configuration Wizard from this installation to create domains, Start menu shortcuts to the domains are not created. In this case, users can manually create shortcuts in their local Start menu folders, if desired. |
| 2 "Local user" | Other users registered on this machine will not have access to the Start menu entries for this installation. |

    If you logged in as a user without Administrator privileges, the Start menu entries are created in your user's local Start menu folder.

    Enter the appropriate number.

11. Enter `next` (or `n`) when you are done.

12. The Installation Summary window shows the products and components you are about to install, along with the approximate size in MB. This window is for your information only; to change the components to be installed, type `Previous` to return to the appropriate window.

13. Enter `next` (or `n`).

    The installer program installs Oracle Event Processing. The Installation Complete window indicates that the product was installed successfully.

14. Type `exit` to exit the program.

15. Review the post-installation steps that Section , "Post-Installation Steps" describes.

# Installing Oracle Event Processing in Silent Mode

This section describes how to install using the Oracle Event Processing installer in silent mode.

Silent-mode installation is a non-interactive method of installing your software that requires the use of an XML properties file for selecting installation options.

For more information, see Section , "Installation Overview".

### To install Oracle Event Processing in silent mode:

1. Log in to the Windows or UNIX computer on which you want to install Oracle Event Processing.

   Be sure you log in to the computer as the user that will be the main administrator of the Oracle Event Processing installation.

2. Download the product distribution file for the platform on which you want to install Oracle Event Processing.

3. Create a `silent.xml` file that defines the configuration settings normally entered by a user during an interactive installation process.

   See Section , "Creating a silent.xml File for Silent-Mode Installation."

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

4. Launch the installation program in silent mode using the commands in the following table appropriate for your platform.

| Platform | Instructions |
|---|---|
| Windows | Open a command window, change to the download directory, and enter the following command:<br>`prompt> filename -mode=silent -silent_xml=path_to_xml_file`<br>In the preceding command, *`filename`* is the name of the installation program specific to your platform (for more information, see http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.html) and *`path_to_xml_file`* is the full pathname of the `silent.xml` template file you created in the preceding step.<br>If you want to create an installation log, use the `-log=`*`full_path_to_log_file`* option; for example:<br>`prompt> filename -mode=silent -silent_xml=path_to_xml_file -log=C:\logs\server_install.log` |

| Platform | Instructions |
|---|---|
| Linux or Solaris | Open a command window, change to the download directory, and enter these commands:<br>`prompt> chmod a+x filename`<br>`prompt> ./filename -mode=silent -silent_xml=path_to_xml_file`<br>In the preceding commands, *filename* is the name of the installation program specific to your platform (for more information, see http://www.oracle.com/technology/software/products/middleware/htdocs/111110_fmw.html) and *path_to_xml_file* is the full pathname of the `silent.xml` template file you created in the preceding step.<br>If you want to create an installation log, use the `-log=full_path_to_log_file` option; for example:<br>`prompt> ./filename -mode=silent -silent_xml=path_to_xml_file -log=C:\logs\server_install.log` |

An Oracle Installer window is displayed, indicating that the files are being extracted. No other prompt or text is displayed.

The installation is complete when the Oracle Installer window disappears.

See Section , "Returning Exit Codes to the Command Window" for getting information about the success or failure of the silent installation.

5. Review the post-installation steps that Section , "Post-Installation Steps" describes.

## Creating a silent.xml File for Silent-Mode Installation

When you install Oracle Event Processing in silent mode, the installation program uses an XML file (`silent.xml`) to determine which installation options should be implemented.

### To create a silent.xml file for silent-mode installation:

1. Using your favorite text editor, create an empty file called `silent.xml` on the computer on which you want to install Oracle Event Processing in silent mode.
2. Copy the contents of the sample XML file, shown in Example 2–1, into your own `silent.xml` file.

***Example 2–1 Sample silent.xml File for Silent-Mode Installation***

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Silent installer option: -mode=silent -silent_xml=C:\oracle\silent.xml -->
<bea-installer>
    <input-fields>
        <data-value name="BEAHOME"            value="C:\oracle_cep" />
        <data-value name="USER_INSTALL_DIR"   value="C:\oracle_cep\ocep_11.1" />
        <data-value name="INSTALL_SHORTCUT_IN_ALL_USERS_FOLDER"   value="yes"/>
        <data-value name="COMPONENT_PATHS" value="Oracle Event Processing" />
    </input-fields>
</bea-installer>
```

3. In the `silent.xml` file you just created, edit the values for the keywords shown in Table 2–3 to reflect your configuration.

   For example, if you want to install into the ORACLE_CEP_HOME directory `e:\oracle_cep`, update the corresponding `<data-value>` element as follows

   ```
   <data-value name="BEAHOME" value="e:\oracle_cep" />
   ```

*Table 2–3 Values for the silent.xml File*

| For this data-value name... | Enter the following value... |
|---|---|
| BEAHOME[1] | The full pathname for the Oracle Middleware Home directory of your choice.<br>For more information, see Section , "Oracle Fusion Middleware Directory Structure and Concepts". |
| USER_INSTALL_DIR | The full pathname for the Oracle Product Home directory for Oracle Event Processing of your choice.<br>For more information, see Section , "Oracle Fusion Middleware Directory Structure and Concepts". |
| INSTALL_SHORTCUT_IN_ALL_USERS_FOLDER | Windows only. Specify:<br>■ true, or yes, to create the shortcuts in the All Users folder.<br>■ false, or no, to create the shortcuts in the local users folder.<br>The user performing the installation must have Administrator privileges to install the Start menu shortcuts in the All Users folder.<br>The default value for this parameter, if you do not specify it, is true. |
| COMPONENT_PATHS | Specify the components and subcomponents of Oracle Event Processing you want to install on your system. Use the following values:<br>■ Oracle Event Processing<br>■ Oracle Event Processing/Event Server<br>■ Oracle Event Processing/Event Server Samples<br>For additional information about entering these values, see Section , "Guidelines for Component Selection."<br>If you do not include the COMPONENT_PATHS data-value name in the silent.xml file, the complete Oracle Event Processing product is installed. |
| LOCAL_JVMS | Option to select supported JVM, which is already installed.<br>**Note:** The presence of the LOCAL_JVMS token negates any default selection and only sets the values assigned for the token as user selection. The value of the token can be a pipe (\|) separated JavaHomes. |

[1] Do not terminate the pathname with a file separator. That is, enter this C:\mydirectory and not C:\mydirectory\.

> **Note:** Silent install does not support the LOCAL_JVMS data-value. To define a local JVM, you must use graphical mode installation as Section , "Installing Oracle Event Processing in Graphical Mode" describes.

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

## Guidelines for Component Selection

Use the following guidelines when you specify values for the COMPONENT_PATHS data-value name:

- When you specify a product component to be installed, all subcomponents that are installed by default in a complete installation are also installed. For example, the following entry installs both Oracle Event Processing and the samples:

```
<data-value name="COMPONENT_PATHS"
    value="Oracle Event Processing" />
```

- To install multiple components or subcomponents, separate the components with a bar (|). Do not leave a space before or after the bar.
- To specify subcomponents, you must specify a component/subcomponent combination for each entry. For example, to explicitly install Oracle Event Processing and the samples, enter the following line in the file:

```
<data-value name="COMPONENT_PATHS" value="Oracle Event Processing/Event
Server|Oracle Event Processing/Event Server Samples" />
```

**Note:** Because this release of Oracle Event Processing includes only the server itself and samples, the preceding example is equivalent to the example in the first bullet.

## Returning Exit Codes to the Command Window

When run in silent mode, the installation program generates exit codes that indicate the success or failure of the installation. These exit codes are shown in Table 2–4.

***Table 2–4 Exit Codes***

| Code | Description |
|---|---|
| 0 | Installation completed successfully |
| -1 | Installation failed due to a fatal error |
| -2 | Installation failed due to an internal XML parsing error |

Example 2–2 provides a sample Windows command file that invokes the installation program in silent mode and echoes the exit codes to the command window from which the script is executed.

***Example 2–2 Sample Windows Command File Displaying Silent-Mode Exit Codes***

```
rem Execute the installer in silent mode
@echo off
ofm_ocep_generic_11.1.1.1.0_32_disk1_1of1.exe -mode=silent -silent_
xml=C:\downloads\silent.xml -log=C:\logs\products_silent.log

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

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

## Uninstalling Oracle Event Processing in Silent Mode

This section describes how to uninstall Oracle Event Processing in silent mode using a script included with the installation. You can run this script in silent mode.

**To install Oracle Event Processing in silent mode:**

1. Log in to the Windows or UNIX computer from which you want to uninstall Oracle Event Processing.

   Be sure you log in to the computer as the user that is the main administrator of the Oracle Event Processing installation.

2. Run the uninstall script using the commands in the following table appropriate for your platform.

| Platform | Instructions |
|---|---|
| Windows | Open a command window and enter the following command:<br>`prompt> $BEA_HOME\ocep_11.1\uninstall\uninstall.cmd -mode=silent`<br>In the preceding command, it is assumed that `$BEA_HOME` has been set, such as during product installation. |
| Linux or Solaris | Open a command window and enter the following command:<br>`prompt> $BEA_HOME/ocep_11.1/uninstall/uninstall.sh -mode=silent`<br>In the preceding command, it is assumed that `$BEA_HOME` has been set, such as during product installation. |

# Installing an Oracle Event Processing Patch

You can download maintenance and security updates for Oracle Event Processing from My Oracle Support. For more information, see *Oracle Smart Update Applying Patches to Oracle WebLogic Server*.

# Post-Installation Steps

After installing Oracle Event Processing:

- If you installed Oracle Event Processing for use with the IBM JVM on IBM AIX (64-bit), then in each Oracle Event Processing domain directory, make the changes that Section , "Configuring Oracle Event Processing for the IBM JDK" describes.
- Try out the product examples.
- Install the Oracle Event Processing IDE for Eclipse as Section , "Installing the Oracle Event Processing IDE for Eclipse" describes.
- Optionally, create your own Oracle Event Processing domain:

  See:

  - "Creating an Oracle Event Processing Standalone-Server Domain" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*
  - "Creating an Oracle Event Processing Multi-Server Domain Using Oracle Coherence" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*
  - "Creating an Oracle Event Processing Multi-Server Domain Using Oracle Event Processing Native Clustering" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*

> **Note:** If you installed Oracle Event Processing using the default option, you must create your own Oracle Event Processing domain. For more information, see Section , "Default Oracle Event Processing Domain ocep_domain and Samples".

- Create an Oracle Event Processing application and deploy it to your domain.

  For a description of the programming model, details about the various components that make up an application, and how they all fit together, see "Overview of Creating Oracle Event Processing Applications" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

## Configuring Oracle Event Processing for the IBM JDK

If you installed Oracle Event Processing for use with the IBM JDK, depending on your operating system and processor architecture, you must make changes to the IBM AIX network options and the `setDomainEnv` script in each Oracle Event Processing domain directory. This section describes:

- Section , "How to Configure Oracle Event Processing for the IBM JVM on IBM AIX (64-bit)"

### How to Configure Oracle Event Processing for the IBM JVM on IBM AIX (64-bit)

If you installed Oracle Event Processing for use with the IBM JDK on IBM AIX (64-bit), you must make changes to the IBM AIX network options and the `setDomainEnv` script in each Oracle Event Processing domain directory.

#### To configure Oracle Event Processing for the IBM JVM on IBM AIX (64-bit):

1. Execute the following commands (using super user privileges or `sudo`) to modify the AIX network options:

   ```
   no -o rfc1323=1
   no -o sb_max=4194304
   ```

2. Go to the domain directory.

   For example, *`MIDDLEWARE_HOME`*`/user_projects/domains/ocep_domain/defaultserver`

3. Edit the `setDomainEnv.sh` script and add the following line:

   ```
   export IBM_JAVA_OPTIONS="-Djava.net.preferIPv4Stack=true
   -Djava.net.preferIPv6Addresses=false"
   ```

# Installing the Oracle Event Processing IDE for Eclipse

Oracle Event Processing IDE for Eclipse is a set of plugins for the Eclipse IDE designed to help develop, deploy, and debug applications for Oracle Event Processing.

For more information, see:

- "Installing the Latest Oracle Event Processing IDE for Eclipse" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*
- "Installing the Oracle Event Processing IDE for Eclipse Distributed With Oracle Event Processing" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*

# Upgrading to Oracle Event Processing 11*g* Release 1 (11.1.1.9)

Upgrading to Oracle Event Processing 11*g* Release 1 (11.1.1.9) is a two-step process: first you must upgrade your applications and then you must upgrade the domain to which the applications are deployed.

Table 2–5 lists the steps you must take for each supported upgrade path:

***Table 2–5 Upgrade Paths***

| From Release | To Release 10.3 | To Release 11gR1 (11.1.1) |
|---|---|---|
| 2.0 | 1. Section , "Upgrading a WebLogic Event Server 2.0 Domain to Oracle Event Processing 10.3"<br>2. Section , "Upgrading a WebLogic Event Server 2.0 Application to Run on Oracle Event Processing 10.3" | 1. Section , "Upgrading a WebLogic Event Server 2.0 Domain to Oracle Event Processing 10.3"<br>2. Section , "Upgrading an Oracle Event Processing 10.3 Domain to Oracle Event Processing 11g Release 1 (11.1.1.9)"<br>3. Section , "Upgrading a WebLogic Event Server 2.0 Application to Run on Oracle Event Processing 10.3"<br>4. Section , "Upgrading an Oracle Event Processing 10.3 Application to Run on Oracle Event Processing 11g Release 1 (11.1.1.9)" |
| 10.3 | Not Applicable. | 1. Section , "Upgrading an Oracle Event Processing 10.3 Domain to Oracle Event Processing 11g Release 1 (11.1.1.9)"<br>2. Section , "Upgrading an Oracle Event Processing 10.3 Application to Run on Oracle Event Processing 11g Release 1 (11.1.1.9)" |

For more information, see Section , "Backward Compatibility Issues".

## Upgrading a WebLogic Event Server 2.0 Domain to Oracle Event Processing 10.3

This section describes the steps you must take to upgrade a WebLogic Event Server 2.0 domain so that it runs correctly in Oracle Event Processing 10.3. For clarity, it is assumed that the existing WebLogic Event Server 2.0 domain is located in the `/bea/user_projects/domains/mydomain20` directory.

### To upgrade a WebLogic Event Server 2.0 domain to Oracle Event Processing 10.3:

1. Using the Configuration Wizard, create a temporary Oracle Event Processing 10.3 domain. Later steps in this procedure require you to use or refer to files in a new Oracle Event Processing 10.3 domain, and it is best to use a new domain. You can later delete this domain if you want.

   For the purposes of this procedure, it is assumed that the new Oracle Event Processing 10.3 domain is called `mydomain30`, it contains a single server called `defaultserver`, and the server files are located in the `/oracle_cep/user_projects/domains/mydomain30/defaultserver` directory.

   See "Creating an Oracle Event Processing Standalone-Server Domain" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.

2. If the WebLogic Event Server 2.0 server is currently running, stop it.

3. Make a backup copy of your WebLogic Event Server 2.0 domain in case you need to revert back.

4. Replace the following two files in the WebLogic Event Server 2.0 domain with the equivalent files from the Oracle Event Processing 10.3 domain.

- lib/XACMLAuthorizerInit.ldift
- lib/XACMLRoleMapperInit.ldift

The WebLogic Event Server 2.0 files are located relative to the domain directory (/bea/user_projects/domains/mydomain20 in our example) and the Oracle Event Processing 10.3 files are located relative to the server directory under the domain directory (/oracle_cep/user_projects/domains/mydomain30/defaultserver in our example).

5. Using your favorite text editor, open the atnstore.txt file in the WebLogic Event Server 2.0 domain, located in the config sub-directory of the main domain directory, and add the new Oracle 10.3 groups:

```
group: wlevsDeployers
description:
group: wlevsApplicationAdmins
description:
group: wlevsBusinessUsers
description:
group: wlevsOperators
description:
```

6. Remove the following files and directories (if they exist) in the WebLogic Event Server 2.0 domain:
   - FileBasedDefaultCredentialMappermy-realmInit.initialized
   - FileBasedXACMLAuthorizermy-realmInit.initialized
   - FileBasedXACMLRoleMappermy-realmInit.initialized
   - rm
   - cm
   - atz
7. Update the startwlevs.cmd (Windows) or startwlevs.sh (Unix) command scripts in the WebLogic Event Server 2.0 domain to point to the new Oracle 10.3 binaries.
8. Update the stopwlevs.cmd (Windows) or stopwlevs.sh (Unix) command scripts in the WebLogic Event Server 2.0 domain to point to the new Oracle 10.3 binaries.
9. Start the server in the 2.0 domain using the Oracle 10.3 binaries.

   "Starting and Stopping an Oracle Event Processing Server in a Standalone-Server Domain" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.
10. This upgrade procedure might have changed the security configuration of your 2.0 domain, especially if you created new users and assigned them to groups. If this is the case, use Visualizer to reconfigure the security.

    See:
    - "Security Tasks" in the *Oracle Fusion Middleware Visualizer User's Guide for Oracle Event Processing*
    - "Configuring Security for Oracle Event Processing" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*

## Upgrading an Oracle Event Processing 10.3 Domain to Oracle Event Processing 11*g* Release 1 (11.1.1.9)

This section describes the steps you must take to upgrade an Oracle Event Processing 10.3 domain so that it runs correctly in Oracle Event Processing 11g Release 1 (11.1.1). For clarity, it is assumed that the existing Oracle Event Processing 10.3 domain is located in the `/bea/user_projects/domains/mydomain103` directory.

**To upgrade an Oracle Event Processing 10.3 domain to Oracle Event Processing release 11*g* Release 1 (11.1.1.9):**

1. Using the Configuration Wizard, create a temporary Oracle Event Processing 11*g* Release 1 (11.1.1.9) domain. Later steps in this procedure require you to use or refer to files in a new Oracle Event Processing 11*g* Release 1 (11.1.1.9) domain, and it is best to use a new domain. You can later delete this domain if you want.

   For the purposes of this procedure, it is assumed that the new Oracle Event Processing 11g Release 1 (11.1.1) domain is called `mydomain11`, it contains a single server called `defaultserver`, and the server files are located in the `/oracle_cep/user_projects/domains/mydomain11/defaultserver` directory.

   See "Creating an Oracle Event Processing Standalone-Server Domain" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.

2. If the WebLogic Event Server 10.3 server is currently running, stop it.

3. Make a backup copy of your WebLogic Event Server 10.3 domain in case you need to revert back.

4. Replace the following two files in the WebLogic Event Server 10.3 domain with the equivalent files from the Oracle Event Processing 11g Release 1 (11.1.1) domain.

   - `lib/XACMLAuthorizerInit.ldift`
   - `lib/XACMLRoleMapperInit.ldift`

   The WebLogic Event Server 10.3 files are located relative to the domain directory (`/bea/user_projects/domains/mydomain30` in our example) and the Oracle Event Processing 11g Release 1 (11.1.1) files are located relative to the server directory under the domain directory (`/oracle_cep/user_projects/domains/mydomain11/defaultserver` in our example).

5. Using your favorite text editor, open the `atnstore.txt` file in the WebLogic Event Server 10.3 domain, located in the `config` sub-directory of the main domain directory, and add the new Oracle 11*g* Release 1 (11.1.1.9) groups:

   ```
   group: wlevsDeployers
   description:
   group: wlevsApplicationAdmins
   description:
   group: wlevsBusinessUsers
   description:
   group: wlevsOperators
   description:
   ```

6. Remove the following files and directories (if they exist) in the WebLogic Event Server 10.3 domain:

   - `FileBasedDefaultCredentialMappermy-realmInit.initialized`
   - `FileBasedXACMLAuthorizermy-realmInit.initialized`
   - `FileBasedXACMLRoleMappermy-realmInit.initialized`

- `rm`
- `cm`
- `atz`

7. Update the `startwlevs.cmd` (Windows) or `startwlevs.sh` (Unix) command scripts in the WebLogic Event Server 10.3 domain to point to the new Oracle 11*g* Release 1 (11.1.1.9) binaries.
8. Update the `stopwlevs.cmd` (Windows) or `stopwlevs.sh` (Unix) command scripts in the WebLogic Event Server 10.3 domain to point to the new Oracle 11*g* Release 1 (11.1.1.9) binaries.
9. Start the server in the 10.3 domain using the Oracle 11*g* Release 1 (11.1.1.9) binaries.

   "Starting and Stopping an Oracle Event Processing Server in a Standalone-Server Domain" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*.
10. This upgrade procedure might have changed the security configuration of your 10.3 domain, especially if you created new users and assigned them to groups. If this is the case, use Visualizer to reconfigure the security.

    See:

    - "Security Tasks" in the *Oracle Fusion Middleware Visualizer User's Guide for Oracle Event Processing*
    - "Configuring Security for Oracle Event Processing" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Event Processing*

## Upgrading a WebLogic Event Server 2.0 Application to Run on Oracle Event Processing 10.3

This section describes the steps you must take to upgrade an application that you developed in Version 2.0 of WebLogic Event Server so that it runs on Oracle Event Processing 10.3.

### To upgrade a WebLogic Event Server 2.0 application to run on Oracle Event Processing 10.3:

1. Update the `MANIFEST.MF` file to import new versions of Spring framework and Oracle Event Processing packages, as well as new required packages. In particular:
   - Update the version of all imported Spring framework packages to 2.5.5. For example:

     ```
     Import-Package:
         org.springframework.aop.framework;version="2.5.5",
         org.springframework.aop;version="2.5.5",
         ...
     ```

   - Update the version of any imported Oracle Event Processing packages to 3.0.0.0. For example:

     ```
     Import-Package:
         com.bea.wlevs.ede;version="3.0.0.0",
         com.bea.wlevs.ede.api;version="3.0.0.0",
         ...
     ```

   - Add the following packages to the Import-Package header if they are not already included:

```
Import-Package:
    com.bea.wlevs.management.configuration.spi;version="3.0.0.0",
    com.bea.wlevs.management.spi;version="3.0.0.0",
    com.bea.wlevs.monitor;version="3.0.0.0",
    com.bea.wlevs.spi;version="3.0.0.0",
    com.bea.wlevs.spring.support;version="3.0.0.0",
    commonj.work;version="1.4.0.0",
    org.springframework.osgi.extensions.annotation;version="1.1.0",
    com.bea.wlevs.ede.spi;version="3.0.0.0",
    com.bea.wlevs.configuration.internal;version="3.0.0.0",
    ...
```

2. If you use Spring or Spring Dynamic Modules for OSGI (Spring DM) features in your application, it is possible that the declaration of the features in the Spring application context file has changed. If this is the case, you must update these declarations in the EPN assembly file of your Oracle Event Processing application.

   **Note:** This change is a result of the upgrade of the Spring framework (from 2.0 to 2.5) that occurred between WebLogic Event Server 2.0 and Oracle Event Processing 10.3, not as a direct result of the Oracle Event Processing upgrade

   Refer to the appropriate 2.5 XSD Schemas for any changes:

   - Spring: http://www.springframework.org/schema/beans/spring-beans.xsd
   - Spring DM: http://www.springframework.org/schema/osgi/spring-osgi.xsd

   The following bullets list some of the typical changes you might have to make; the following list is not complete:

   - When specifying a property to the `<osgi:service-property>` tag, use the `<entry>` tag with the key and value attributes, rather than the old `<prop>` tag.

     For example, change the following 2.0 tag from:

     ```
     <osgi:service-properties>
         <prop key="type">SocketAdapterType</prop>
     </osgi:service-properties>
     ```

     To:

     ```
     <osgi:service-properties>
         <entry key="type" value="SocketAdapterType"/>
     </osgi:service-properties>
     ```

   - The `value` or `ref` attribute of an `instance-property` must always be set to an explicit value; it can no longer be an empty string to indicate an implicit use of a default value.

     For example, change the following 2.0 tag from:

     ```
     <wlevs:adapter id="fileAdapter" provider="FileAdapterType">
         <!-- file: empty value uses default
         <wlevs:instance-property name="file" value="" />
         <wlevs:listener ref="algoTradingProcessor"/>
     </wlevs:adapter>
     ```

     To:

```
<wlevs:adapter id="fileAdapter" provider="FileAdapterType">
    <wlevs:instance-property name="file" value="test.file" />
    <wlevs:listener ref="algoTradingProcessor"/>
</wlevs:adapter>
```

3. Recompile the Java code of your 2.0 adapter and business POJO implementations using your IDE. If you get compile-time errors, check the latest 10.3 Javadoc (`http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/javadocs/wlevs/index.html`) that describe the new Oracle Event Processing APIs and make the appropriate source code changes.
4. If your 2.0 application has an adapter that uses the `loadgen` provider as Example 2–3 shows, then you must register a `StockTick` event type in your EPN assembly file as Example 2–4 shows.

***Example 2–3 Adapter Using loadgen Provider***

```
<wlevs:adapter id="fxMarketAmer" provider="loadgen">
    <wlevs:instance-property name="port" value="9011"/>
</wlevs:adapter>
```

***Example 2–4 Registering a StockTick Event***

```
<wlevs:event-type-repository>
    <wlevs:event-type type-name="StockTick">
        <wlevs:class>com.bea.wlevs.adapter.defaultprovider.StockTickEvent</wlevs:c
lass>
    </wlevs:event-type>
</wlevs:event-type-repository>
```

5. After you have made the preceding changes, reassemble the application and deploy it to Oracle Event Processing 10.3.

   See "Assembling and Deploying Oracle Event Processing Applications" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

   If, during deployment, you get an exception that indicates that a package is invisible, add this package to the Import-Package header of the `MANIFEST.MF` file, then reassemble and redeploy the application. Keep adding packages in this manner until the application deploys successfully.

## Upgrading an Oracle Event Processing 10.3 Application to Run on Oracle Event Processing 11*g* Release 1 (11.1.1.9)

This section describes the steps you must take to upgrade an application that you developed in Oracle Event Processing 10.3 so that it runs on Oracle Event Processing 11*g* Release 1 (11.1.1.9).

**To upgrade an Oracle Event Processing 10.3 application to run on Oracle Event Processing release 1111*g* Release 1 (11.1.1.9):**

1. Update the `MANIFEST.MF` file to import new versions of Spring framework and Oracle Event Processing packages, as well as new required packages.

   Note that alternatively you can specify unversioned packages which will not require updating and also that you can specify larger versions in order to avoid minor version updates, that is, use "2.5" instead of "2.5.6".

   In particular:

- Update the version of all imported Spring framework packages to 2.5.6. For example:

```
Import-Package:
    org.springframework.aop.framework;version="2.5.6",
    org.springframework.aop;version="2.5.6",
    ...
```

- Update the version of all imported Spring-DM framework packages to 1.2.0. For example:

```
Import-Package:
    org.springframework.osgi.context="1.2.0",
    ...
```

- Update the version of any imported Oracle Event Processing packages to 11.1.1.4_0. For example:

```
Import-Package:
    com.bea.wlevs.ede;version="11.1.1.4_0",
    com.bea.wlevs.ede.api;version="11.1.1.4_0",
    ...
```

- Add the following packages to the Import-Package header if they are not already included (see the sample source for a complete list of headers that may be required):

```
Import-Package:
    com.bea.wlevs.management.spi;version="11.1.1.4_0",
    com.bea.wlevs.spring.support;version="11.1.1.4_0",
    com.bea.wlevs.ede.spi;version="11.1.1.4_0",
    org.springframework.osgi.extensions.annotation;version="1.2.0",
    ...
```

2. If you use Spring or Spring Dynamic Modules for OSGI (Spring DM) features in your application, it is possible that the declaration of the features in the Spring application context file has changed. If this is the case, you must update these declarations in the EPN assembly file of your Oracle Event Processing application.

> **Note:** This change is a result of the upgrade of the Spring-DM framework (from 1.1 to 1.2) that occurred between Oracle Event Processing 10.3 and Oracle Event Processing 11.1, not as a direct result of the Oracle Event Processing upgrade.

Refer to the appropriate 2.5 XSD Schemas for any changes:

- Spring: http://www.springframework.org/schema/beans/spring-beans.xsd
- Spring DM: http://www.springframework.org/schema/osgi/spring-osgi.xsd

In particular convert any Spring-DM declared adapter factories to use the `<wlevs:factory/>` tag instead. For example, if your 10.3 EPN assembly file contains the `service` that Example 2–5 shows, then you must replace this service with the `wlevs:factory` that Example 2–6 shows.

***Example 2–5 Spring-DM Declared Adapter Factory***

```
<osgi:service interface="com.bea.wlevs.ede.api.AdapterFactory">
    <osgi:service-properties>
```

```
        <entry key="type" value="SocketAdapterType"/>
    </osgi:service-properties>
    <bean class="com.bea.wlevs.example.algotrading.adapter.SocketAdapterFactory"
/>
</osgi:service>
```

***Example 2–6 wlevs:factory***

```
<wlevs:factory provider-name="SocketAdapterType"
    class="com.bea.wlevs.example.algotrading.adapter.SocketAdapterFactory"/>
```

3. Recompile the Java code of your 10.3 adapter and business POJO implementations using your IDE. If you get compile-time errors, check the latest 11g Release 1 (11.1.1) Javadoc (see *Oracle Fusion Middleware Java API Reference for Oracle Event Processing*) that describe the new Oracle Event Processing APIs and make the appropriate source code changes.
4. After you have made the preceding changes, reassemble the application and deploy it to Oracle Event Processing 11g Release 1 (11.1.1).

   See "Assembling and Deploying Oracle Event Processing Applications" in the *Oracle Fusion Middleware Developer's Guide for Oracle Event Processing for Eclipse*.

   If, during deployment, you get an exception that indicates that a package is invisible, add this package to the Import-Package header of the `MANIFEST.MF` file, then reassemble and redeploy the application. Keep adding packages in this manner until the application deploys successfully.

## Backward Compatibility Issues

The following are non-backward compatible changes in the management framework:

- The following classes have been deprecated and removed from all operation signatures:
  - `com.bea.wlevs.management.ManagementException`
  - `com.bea.wlevs.management.ManagementRuntimeException`
  - `com.bea.wlevs.management.MbeanOperationsException`
- The following methods have been removed from all MBeans: `isRegistered()`, `preRegister()`, `postRegister()`, `getMBeanInfo()`.
- The monitoring-related methods have been removed from `StageMBean` and replaced by `com.bea.wlevs.monitor.management.MonitorRuntimeMBean`.
- The `com.bea.wlevs.management.boot.BootMBean` has been removed.
- The `com.bea.wlevs.management.configuration.ConfigSessionBean` has been removed.
- The `ObjectName` for the `AppDeploymentMBean` has been changed to include the `DomainMBean` as a parent.
- The class `com.bea.wlevs.server.management.mbean.ServerRuntimeMBean` has been changed to `com.bea.wlevs.management.runtime.ServerRuntimeMBean`.
- Two additional modules have been added: `com.bea.wlevs.management.api_*` and `com.bea.wlevs.management.spi_*`, in addition to the existing `com.bea.wlevs.management_*`.
- The service `com.bea.wlevs.spi.ManagementService` has been moved from bundle `com.bea.wlevs.spi_*` to `com.bea.wlevs.management.spi_*`.

# Glossary

### Adapter

An element of the **EPN** that interfaces directly to an inbound event source. Adapters understand the inbound protocol, and are responsible for converting the event data into a normalized form that can be queried by a **POJO**. Adapters forward the normalized event data into a **Stream**.

### Aggregate Function

Aggregate functions return a single aggregate result based on group of tuples, rather than on a single tuple.

See also **Function** and **Single-Row Function**.

### OEP

Oracle Event Processing.

### Channel

A channel represents the physical conduit through which events flow between other types of components, such as between an **Adapter** and a **Processor**, and between a **Processor** and an **Event Bean**. A channel can model a **Stream** or **Relation**.

### Condition

An Oracle CQL condition specifies a combination of one or more expressions and logical (Boolean) operators and returns a value of `TRUE`, `FALSE`, or `UNKNOWN`.

### Constant value

A fixed data value. Synonymous with **Literal**.

### CQL

Oracle Continuous Query Language. Supersedes **EPL**.

### Data Feed

A synonym for **Event Source**.

### Destination

An Oracle CQL destination identifies a consumer of query results such as the Enterprise Link BAM Adapter, JMS queue or topic, or file.

### Deterministic Garbage Collection

Short, predictable pause times for memory heap garbage collection, which is the process of clearing dead objects from the heap, thus releasing that space for new objects.

### DStream

A relation-to-stream operator that represents deleted tuples.

### EDA

Event-Driven Architecture.

### EPL

Oracle Event Processing Language. Superseded by **CQL**.

### EPN

Oracle Event Processing Network. An EPN is the arbitrary interconnection of **Adapter**, **Stream**, **POJO**, and business logic POJOs used by Oracle Event Processing to process events.

### Event Bean

A **POJO** to that contains the business logic executed when a notable event is detected. An event bean is an **Event Sink**.

### Event Rule

A query, expressed in **CQL** or **EPL**, executed by a **POJO** to filter and aggregate events.

### Event Sink

A component that consumes events, such as a **Processor**.

See also **Event Source**.

### Event Source

A component that provides events, such as a sensor, wire service, or stock ticker.

See also **Data Feed** and **Event Sink**.

### Expressions

An Oracle CQL expression is a combination of one or more values, operators, and Oracle CQL functions that evaluates to a value. An expression generally assumes the datatype of its components.

See also **Condition** and **Function**.

### Format model

A character **literal** that describes the format of datetime or numeric data stored in a character string.

### Function

Oracle CQL functions are similar to operators in that they manipulate data items and return a result. Functions differ from operators in the format of their arguments. This format enables them to operate on zero, one, two, or more arguments.

See also **Condition**, **Aggregate Function**, and **Single-Row Function**.

### Incremental Processing

A user-defined aggregate function design pattern that improves scalability and performance by ensuring that the cost of (re)computation on arrival of new events will be proportional to the number of new events as opposed to the total number of events seen thus far.

If your user-defined aggregate function supports incremental processing, you specify the `supports incremental processing` clause in the `register function` statement to instruct the Oracle Event Processing Service Engine to supply only the new event data as opposed to performing a rescan over already processed event data.

### IStream

A relation-to-stream operator that represents inserted tuples.

### Join

A query that combines rows from two or more streams, views, or relations.

### Latency

An expression of how much time it takes for data to get from one designated point to another.

### Literal

A fixed data value. Synonymous with **Constant value**.

### Monotonic

A sequence of values that are consistently increasing and never decreasing or consistently decreasing and never increasing. The sequence may contain multiple consecutive occurrences of the same value.

### Now window

A special case of the time-based sliding window on a stream `S` that takes a time-interval `T` as a parameter and is specified by: `S [Range T]`. A `Now` window is defined as: `S [Now]` (short for `S [Range 0]`). When `T = 0`, the relation at time `t` consists of tuples obtained from elements of `S` with timestamp `t`.

See also **Sliding window**.

### Operators

Oracle CQL operators manipulate data items and return a result. Syntactically, an operator appears before or after an operand or between two operands.

### OSGi

A dynamic module system for Java that provides a service-oriented, component-based environment and standardized software lifecycle management. Oracle Event Processing applications are packaged and deployed as OSGi bundles. For more information, see `http://www.osgi.org/`.

### Partitioned window

A partitioned sliding window on a stream `S` takes a positive integer number of tuples `N` and a subset `{A1,... Ak}` of the stream's attributes as parameters and is specified by: `S[Partition By A1 ... Ak Rows N]` or, optionally, `S[Partition By A1 ... Ak Rows N Range T]`.

See also **Sliding window**.

### POJO

A Plain Old Java Object. A Java class that is not required to implement a third-party interface or extend a third-party class. In Oracle Event Processing, you can express your business logic using POJOs.

### Processor

An element of the **EPN** that consumes normalized event data from a stream, processes it using queries (expressed in **CQL** or **EPL**), and may generate new events to an output stream.

### Query

A query is an operation that retrieves data from one or more streams or views. In this reference, a top-level `SELECT` statement is called a **query**.

### Real-time

A level of computer responsiveness that a user senses as sufficiently immediate or that enables the computer to keep up with some external process (for example, to present visualizations of the weather as it constantly changes).

### Relation

A relation is time-varying bag of tuples. Here "time" refers to an instant in the time domain. At every instant of time, a relation is a bounded set. It can also be represented as a sequence of timestamped tuples that includes insertions, deletions, and updates to capture the changing state of the relation. The updates are required to arrive at the system in the order of increasing timestamps. Like streams, relations have a fixed schema to which all tuples conform.

### RStream

A relation-to-stream operator that maintains the entire current state of its input relation and outputs all of the tuples as insertions at each time step.

### Single-Row Function

Single-row functions return a single result row for every row of a queried stream or view.

See also **Function** and **Aggregate Function**.

### Sliding window

A stream-to-relation operator based on the window specification derived from SQL99.

See also: **Now window**, **Partitioned window**, **Unbounded window, tuple-based**, and **Unbounded window, time-based**.

### Source

An Oracle CQL source identifies a producer of data that a Oracle CQL query operates on such as the Enterprise Link BAM Adapter, JMS queue or topic, or file.

### Spring Framework

A light-weight, open source application framework for Java. Oracle Event Processing server uses the Spring Framework to host Oracle Event Processing applications. For more information, see `http://www.springframework.org/`.

### Stream

A stream is a sequence of timestamped tuples. There could be more than one tuple with the same timestamp. The tuples of an input stream are required to arrive at the system in the order of increasing timestamps. A stream has an associated schema consisting of a set of named attributes, and all tuples of the stream conform to the schema.

A stream is a bag (or multi-set) of tuple-timestamp pairs, which can be represented as a sequence of timestamped tuple "insertions".

In Oracle Event Processing, a stream is modeled as a channel component.

See also **Tuple** and **Channel**.

### Throughput

An Oracle CQL source identifies a producer of data that a Oracle CQL query operates on such as the Enterprise Link BAM Adapter, JMS queue or topic, or file.

### Tuple

The term "tuple of a stream" denotes the ordered list of data (excluding timestamp data) portion of a stream element (the `s` of `<s,t>`). For example, a stock ticker data stream might appear like this where each stream element is made up of `<timestamp value>`, `<stock symbol>`, and `<stock price>`:

```
...
<timestampN>    NVDA,4
<timestampN+1>  ORCL,62
<timestampN+2>  PCAR,38
<timestampN+3>  SPOT,53
<timestampN+4>  PDCO,44
<timestampN+5>  PTEN,50
...
```

In the stream element `<timestampN+1>  ORCL,62`, the tuple is `ORCL,62`.

See also **Stream**.

### Unbounded window, time-based

A special case of the time-based sliding window on a stream `S` that takes a time-interval `T` as a parameter and is specified by: `S [Range T]`. An `Unbounded` window is defined as: `S [Range Unbounded]` (short for `S [Range infinity]`). When `T = infinity`, the relation at time `t` consists of tuples obtained from all elements of `S` up to `t`.

See also **Sliding window**.

### Unbounded window, tuple-based

A special case of the tuple-based sliding window on a stream `S` that takes a number of tuples `N` as a parameter and is specified by: `S [Rows N]`. An `Unbounded` window is defined as: `S [Rows Unbounded]` (short for `S [Rows infinity]` and equivalent to `S [Range Unbounded]`). When `T = infinity`, the relation at time `t` consists of tuples obtained from all elements of `S` up to `t`.

See also **Sliding window**.

### View

An Oracle CQL view represents an alternative selection on a stream or relation. In Oracle CQL, you use a view instead of a subquery.