Oracle® Fusion Middleware

Schema Reference for Oracle Event Processing

12*c* Release (12.1.3)

E53667-05

November 2016

How to design and create Oracle Event Processing scalable applications to process streaming events.

ORACLE

Oracle Fusion Middleware Schema Reference for Oracle Event Processing, 12*c* Release (12.1.3)

E53667-05

Primary Author: Oracle Corporation

Contents

Preface

This document provides a reference to Oracle Event Processing schemas. When you install Oracle Event Processing, the schema files are installed in the following directory:

`/Oracle/Middleware/my_oep/xsd`

Audience

This document is intended for developers who want to create Oracle Event Processing applications.

Related Documents

For more information, see the following:

- Known Issues for Oracle SOA Products and Oracle AIA Foundation Pack at: http://www.oracle.com//technetwork/middleware/soasuite/documentation/soaknown-2644661.html.
- *Developing Applications for Oracle Event Processing*
- *Getting Started with Oracle Event Processing*
- *Administering Oracle Event Processing*
- *Using Visualizer for Oracle Event Processing*
- *Customizing Oracle Event Processing*
- *Developing Applications with Oracle CQL Data Cartridges*
- *Oracle CQL Language Reference for Oracle Event Processing*
- *Java API Reference for Oracle Event Processing*
- *Using Oracle Stream Explorer*
- *Getting Started with Oracle Stream Explorer*
- *Oracle Database SQL Language Reference* at: `http://docs.oracle.com/cd/E16655_01/server.121/e17209/toc.htm`
- SQL99 Specifications (ISO/IEC 9075-1:1999, ISO/IEC 9075-2:1999, ISO/IEC 9075-3:1999, and ISO/IEC 9075-4:1999)

- Oracle Event Processing Forum: http://forums.oracle.com/forums/forum.jspa?forumID=820

Conventions

The following text conventions are used in this document:

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

What's New in This Guide

The product has been renamed from Oracle Stream Explorer to Oracle Stream Analytics in the 12*c* (12.2.1.0.0) release. The file names, screenshots, and text is updated to reflect new file names and file system structure.

Screens shown in this guide may differ from your implementation, depending on the skin used. Any differences are cosmetic.

The support for QuickFix Adapter has been deprecated in this release.

1

About XML Schema Definitions

Oracle Event Processing uses XML schema definitions (xsd's) to describe the types that you can use in XML documents that define application, debugging, deployment, component, server, data cartridge, and binding configurations. The Oracle Event Processing modules in Oracle JDeveloper generate these configurations according to the xsd's described in this book.

The Oracle Event Processing schema files are provided in the installation in the */Oracle/Middleware/my_osa/xsd* directory.

This chapter includes the following sections:

- EPN Assembly Schema spring-wlevs-v12_1_3_0.xsd
- Component Configuration Schema wlevs_application_config.xsd
- Deployment Schema deployment.xsd
- Server Configuration Schema wlevs_server_config.xsd.

1.1 EPN Assembly Schema spring-wlevs-v12_1_3_0.xsd

You use the EPN assembly file to declare the components that make up your Oracle Event Processing application and how they connect to each other. The EPN assembly file is an extension of the standard Spring context file. You also use the file to register the Java classes that implement the adapter and POJO components of your application, register the event types that you use throughout your application and EPL rules, and reference in your environment the Oracle Event Processing-specific services.

The following XML file shows the EPN assembly file for the HelloWorld example:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:osgi="http://www.springframework.org/schema/osgi"
       xmlns:wlevs="http://www.bea.com/ns/wlevs/spring"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/osgi
           http://www.springframework.org/schema/osgi/spring-osgi.xsd
           http://www.bea.com/ns/wlevs/spring
           http://www.bea.com/ns/wlevs/spring/spring-wlevs-v12_1_3_0.xsd">

    <wlevs:event-type-repository>
        <wlevs:event-type type-name="HelloWorldEvent">
            <wlevs:class>com.bea.wlevs.event.example.helloworld.HelloWorldEvent</
wlevs:class>
        </wlevs:event-type>
    </wlevs:event-type-repository>
```

```
<!-- Adapter can be created from a local class, without going through a adapter
factory -->
    <wlevs:adapter id="helloworldAdapter"
class="com.bea.wlevs.adapter.example.helloworld.HelloWorldAdapter" >
        <wlevs:instance-property name="message" value="HelloWorld - the current time
is:"/>
    </wlevs:adapter>

    <wlevs:channel id="helloworldInputChannel" event-type="HelloWorldEvent" >
        <wlevs:listener ref="helloworldProcessor"/>
        <wlevs:source ref="helloworldAdapter"/>
    </wlevs:channel>

    <!-- The default processor for Oracle Event Processing 12.1.3.0 is CQL -->
    <wlevs:processor id="helloworldProcessor" />

    <wlevs:channel id="helloworldOutputChannel" event-type="HelloWorldEvent"
advertise="true">
        <wlevs:listener>
            <bean class="com.bea.wlevs.example.helloworld.HelloWorldBean"/>
        </wlevs:listener>
        <wlevs:source ref="helloworldProcessor"/>
    </wlevs:channel>
</beans>
```

1.2 Component Configuration Schema wlevs_application_config.xsd

An Oracle Event Processing application contains one or more component configuration files in its `META-INF/wlevs` directory. You use component configuration files to override the default configuration for Oracle Event Processing components such as adapters, channels, and processors. The `wlevs_application_config.xsd` schema file describes the structure of component configuration files. This XSD schema imports the following schemas:

- `wlevs_base_config.xsd`
- `wlevs_eventstore_config.xsd`
- `wlevs_diagnostic_config.xsd`

The following component configuration file is for the `HelloWorld` sample application:

```
<?xml version="1.0" encoding="UTF-8"?><n1:config xmlns:n1="http://www.bea.com/ns/
wlevs/config/application"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <processor>
        <name>helloworldProcessor</name>
        <rules>
            <query id="helloworldRule">
                <![CDATA[ select * from helloworldInputChannel [Now] >
            </query>
        </rules>
    </processor>
    <channel>
        <name>helloworldInputChannel</name>
        <max-size>10000</max-size>
        <max-threads>2</max-threads>
    </channel>
    <channel>
```

```
        <name>helloworldOutputChannel</name>
        <max-size>10000</max-size>
        <max-threads>2</max-threads>
    </channel>
</n1:config>
```

1.3 Deployment Schema deployment.xsd

The `deployments.xml` file is in the `/Oracle/Middleware/my_osa/user_projects/domains/<domain>/<server>/` directory. This XML file lists the OSGi bundles that have been deployed to the server. The `deployment.xsd` schema file describes the structure of deployment files.

The following example shows the `deployments.xml` file for the sample FX domain:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wlevs="http://www.bea.com/ns/wlevs/deployment"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd
  http://www.bea.com/ns/wlevs/deployment
  http://www.bea.com/ns/wlevs/deployment/deployment.xsd">
  <bean
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
      <property name="systemPropertiesModeName"
value="SYSTEM_PROPERTIES_MODE_OVERRIDE"/>
  </bean>
  <wlevs:deployment id="fx" state="start"
    location="file:${wlevs.domain.home}/applications/fx/
com.bea.wlevs.example.fx_11.1.0.0.jar"/>
</beans>
```

1.4 Server Configuration Schema wlevs_server_config.xsd

The Oracle Event Processing server configuration file is located in the *`DOMAIN_DIR`*`/`*`servername`*`/config` directory To change the configuration of an Oracle Event Processing instance, update this file manually to add or remove server configuration elements. The `welvs_server_config.xsd` schema file describes the structure of server configuration files.

The following example shows how to configure some of these services:

```
<?xml version="1.0" encoding="UTF-8"?>
<n1:config xsi:schemaLocation="
    http://www.bea.com/ns/wlevs/config/server wlevs_server_config.xsd"
 xmlns:n1="http://www.bea.com/ns/wlevs/config/server"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <netio>
    <name>NetIO</name>
    <port>9002</port>
  </netio>
  <netio>
    <name>sslNetIo</name>
    <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
    <port>9003</port>
  </netio>
  <work-manager>
    <name>JettyWorkManager</name>
    <min-threads-constraint>5</min-threads-constraint>
```

```
  <max-threads-constraint>10</max-threads-constraint>
</work-manager>
<jetty>
  <name>JettyServer</name>
  <network-io-name>NetIO</network-io-name>
  <work-manager-name>JettyWorkManager</work-manager-name>
  <secure-network-io-name>sslNetIo</secure-network-io-name>
</jetty>
<rmi>
  <name>RMI</name>
  <http-service-name>JettyServer</http-service-name>
</rmi>
<jndi-context>
 <name>JNDI</name>
</jndi-context>
<exported-jndi-context>
  <name>exportedJndi</name>
  <rmi-service-name>RMI</rmi-service-name>
</exported-jndi-context>
<jmx>
  <rmi-service-name>RMI</rmi-service-name>
  <jndi-service-name>JNDI</jndi-service-name>
</jmx>
<ssl>
  <name>sslConfig</name>
  <key-store>./ssl/dsidentity.jks</key-store>
  <key-store-pass>
      <password>changeit</password>
  </key-store-pass>
  <key-store-alias>ds</key-store-alias>
  <key-manager-algorithm>SunX509</key-manager-algorithm>
  <ssl-protocol>TLS</ssl-protocol>
  <enforce-fips>false</enforce-fips>
  <need-client-auth>false</need-client-auth>
</ssl>
<http-pubsub>
  <name>pubsub</name>
  <path>/pubsub</path>
  <pub-sub-bean>
    <server-config>
      <name>pubsubbean</name>
      <supported-transport>
        <types>
          <element>long-polling</element>
        </types>
      </supported-transport>
      <publish-without-connect-allowed>true</publish-without-connect-allowed>
    </server-config>
  <channels>
      <element>
        <channel-pattern>/evsmonitor</channel-pattern>
      </element>
      <element>
        <channel-pattern>/evsalert</channel-pattern>
      </element>
      <element>
        <channel-pattern>/evsdomainchange</channel-pattern>
      </element>
    </channels>
  </pub-sub-bean>
```

```
  </http-pubsub>
</n1:config>
```

2

Application Assembly Schema

The application assembly schema is behind the assembly file where you declare EPN components. This chapter provides a reference to the elements of the `spring-wlevs-v12_1_3_0.xsd` schema.

This chapter includes the following sections:

- wlevs:property
- wlevs:source
- wlevs:table
- wlevs:table-source.

2.1 Application Assembly Element Hierarchy

Oracle Event Processing provides a number of application assembly elements that you use in the EPN assembly file of your application to register event types, declare the components of the event processing network and specify how they are linked together. The EPN assembly file is an extension of the standard Spring context file.

The Oracle Event Processing application assembly elements are organized into the following hierarchy:

```
beans
    Standard Spring and OSGi elements such as bean, osgi-service, and so on.
    wlevs:event-type-repository
        wlevs:event-type
            wlevs:class
            wlevs:metadata
            wlevs:properties
            wlevs:property
    wlevs:adapter
        wlevs:listener
        wlevs:instance-property
        wlevs:property
    wlevs:processor
        wlevs:listener
        wlevs:source
        wlevs:function
        wlevs:instance-property
        wlevs:property
        wlevs:cache-source
        wlevs:table-source
    wlevs:channel
        wlevs:listener
        wlevs:source
        wlevs:instance-property
        wlevs:property
        wlevs:application-timestamped
            wlevs:expression
    wlevs:event-bean
        wlevs:listener
        wlevs:instance-property
        wlevs:property
    wlevs:factory
    wlevs:cache
        wlevs:caching-system
        wlevs:cache-loader
        wlevs:cache-store
        wlevs:cache-listener
    wlevs:caching-system
        wlevs:instance-property
```

wlevs:property
wlevs:table

2.2 wlevs:adapter

Use the `wlevs:adapter` element to declare an adapter component to the Spring application context.

Child Elements

The `wlevs:adapter` application assembly element supports the following child elements:

- wlevs:listener
- wlevs:instance-property

  wlevs:property.

Attributes

Table 2-1 Attributes of the wlevs:adapter Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this adapter, if one exists.	`String`	Yes
`advertise`	Advertises this service in the OSGi registry. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No
`listeners`	Specifies the components that listen to this component. Set this attribute to the value of the `id` attribute of the element that declared the component.	`String`	No

Table 2-1 (Cont.) Attributes of the wlevs:adapter Application Assembly Element

Attribute	Description	Data Type	Required?

`provider`	Specifies the adapter service provider. Typically the value of this attribute is a reference to the OSGi-registered adapter factory service.
	If you are using the `csvgen` or `loadgen` utilities to simulate a data feed, use the hard-coded `csvgen` or `loadgen` values, respectively, such as:
	`provider="csvgen"`
	If you are using one of the built-in HTTP publish-subscribe adapters, then specify the following hard-coded values:
	• For the built-in pub-sub adapter used for *publishing*, specify the hard-coded `httppub` value, such as:
	`provider="httppub"`
	• For the built-in pub-sub adapter used for *subscribing*, specify the hard-coded `httpsub` value, such as:
	`provider="httpsub"`
	If you are using a JMS adapter, then specify one of the following hard-coded values:
	• For the inbound JMS adapter, specify the `jms-inbound` value, such as:
	`provider="jms-inbound"`
	• For the outbound JMS adapter, specify the `jms-outbound` value, such as:
	`provider="jms-outbound"`
	You must specify either the `provider` or `class` attribute, but not both, otherwise an exception is raised.
		`String`	No

`class`	Specifies the Java class that implements this adapter. You must specify either the `provider` or `class` attribute, but not both, otherwise an exception is raised.	`String`	No
`onevent-method`	Specifies the method of the adapter implementation that corresponds to the life cycle `onEvent` method. Oracle Event Processing invokes this method when the adapter receives an event.	`String`	No

Table 2-1 (Cont.) Attributes of the wlevs:adapter Application Assembly Element

Attribute	Description	Data Type	Required?
`init-method`	Specifies the method of the adapter implementation that corresponds to the life cycle `init` method. Oracle Event Processing invokes this method after it has set all the supplied instance properties. This method allows the adapter instance to perform initialization only possible when all bean properties have been set and to throw an exception in the event of mis-configuration.	`String`	No
`activate-method`	Specifies the method of the adapter implementation that corresponds to the life cycle `activate` method. Oracle Event Processing invokes this method after the dynamic configuration of the adapter has completed. This method allows the adapter instance to perform initialization only possible when all dynamic bean properties have been set and the EPN has been wired.	`String`	No
`suspend-method`	Specifies the method of the adapter implementation that corresponds to the life cycle `suspend` method. Oracle Event Processing invokes this method when the application is suspended.	`String`	No
`destroy-method`	Specifies the method of the adapter implementation that corresponds to the life cycle `destroy` method. Oracle Event Processing invokes this method when the application is stopped.	`String`	No

Example

The following example shows how to use the `wlevs:adapter` element in the EPN assembly file. In the example, the adapter's unique identifier is `helloworldAdapter`. The provider is an OSGi service, also registered in the EPN assembly file, whose reference is `hellomsgs`. The adapter has a static property called `message`, which implies that the adapter Java file has a `setMessage` method.

```
<wlevs:adapter id="helloworldAdapter" provider="hellomsgs">
    <wlevs:instance-property name="message"
                             value="HelloWorld - the current time is:"/>
</wlevs:adapter>
```

2.3 wlevs:application-timestamped

Use this element to specify if an wlevs:channel is application time stamped, that is, if the application is responsible for assigning a time stamp to each event, using any time domain.

Otherwise, wlevs:channel is system time stamped, that is, the Oracle Event Processing server is responsible for assigning a time stamp to each event using `System.nanoTime`.

Child Elements

The `wlevs:application-timestamped` application assembly element supports the following child elements.

- wlevs:expression: Specifies an expression to be used as an application time stamp for event processing.

Attributes

Table 2-2 Attributes of the wlevs:application-timestamped Application Assembly Element

Attribute	Description	Data Type	Required?
`is-total-order`	When `true`, indicates that the application time published is always strictly greater than the last value used. Valid values are `true` or `false`. Default: `false`.	`Boolean`	No
`is-silent-relation`	When `true`, indicates that a relation is silent. A silent relation does not emit changes frequently. This setting applies only when the wlevs:channel element attribute `is-relation` is `true`. Valid values are `true` and `false`. Default is `false`.	`Boolean`	No

Example

The following example shows how to use the `wlevs:application-timestamped` element in the EPN assembly file to specify an implicitly application time stamped channel. In the example, the application handles event time stamps internally.

```
<wlevs:channel id="fxMarketAmerOut" >
    <wlevs:application-timestamped>
    </wlevs:application-timestamped>
</wlevs:channel>
```

The following example shows how to use `wlevs:application-timestamped` element in the EPN assembly file to specify an explicitly application time stamped channel by specifying the `wlevs:expression` element. In the example, the `wlevs:expression` element defines the arithmetic expression used to assign a time stamp to each event.

```
<wlevs:channel id="fxMarketAmerOut" >
    <wlevs:application-timestamped>
        <wlevs:expression>mytime+10</wlevs:expression>
    </wlevs:application-timestamped>
</wlevs:channel>
```

The following example adds the `is-silent-relation` and `is-relation` attributes set to true. These settings mean that the channel does not emit changes very frequently.

```
<wlevs:channel id="AppTimeStampedChannel"/>
   <wlevs:application-timestamped is-silent-relation="true" is-relation="true">
   <wlevs:expression>mytime+10</wlevs:expression>
</wlevs:application-timestamped>
```

2.4 wlevs:cache

Use this element to declare a cache to the Spring application context.

Child Elements

The `wlevs:cache` application assembly element supports the following child elements.

- `wlevs:caching-system`—Specifies the caching system to which this cache belongs.

  Note:

  This child element differs from the wlevs:caching-system element used to *declare* a caching system. The child element of the `wlevs:cache` element takes a single attribute, `ref`, that *references* the `id` attribute of a declared caching system.

- wlevs:cache-loader—Specifies the cache loader for this cache.
- wlevs:cache-store—Specifies a cache store for this cache.
- wlevs:cache-listener—Specifies a listener for this cache, or a component to which the cache sends events.

Attributes

Table 2-3 Attributes of the wlevs:cache Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this cache.	`String`	Yes.
`name`	Specifies an alternate name for this cache. If not specified, then the name of the cache is the same as its `id` attribute.	`String`	No.

Table 2-3 (Cont.) Attributes of the wlevs:cache Application Assembly Element

Attribute	Description	Data Type	Required?
`key-properties`	Specifies a comma-separated list of property names that form the unique key value for the objects in the cache, or *cache key*. A cache key can be composed of a single property or multiple properties. When you configure a cache as a listener in an event processing network, Oracle Event Processing inserts events that reach the cache using the unique key value as a key. If you specify a key class with the `key-class` attribute, then this attribute is optional. If you specify neither `key-properties` nor `key-class`, then Oracle Event Processing uses the event object itself as both the key and value when it inserts the event object into the cache.	`String`	No.
`key-class`	Specifies the name of the Java class used for the cache key when the key is a composite key. If you do not specify the `key-properties` attribute, then all properties on the `key-class` are assumed to be key properties. If you specify neither `key-properties` nor `key-class`, then Oracle Event Processing uses the event object itself as both the key and value when it inserts the event object into the cache	`String`	No.
`value-type`	Specifies the type for the values contained in the cache. Must be a valid type name in the event type repository. This attribute is required only if the cache is referenced in an Oracle CQL query. This is because the query processor needs to know the type of events in the cache.	`String`	No.
`caching-system`	Specifies the caching system where this cache is contained. The value of this attribute corresponds to the `id` attribute of the appropriate `wlevs:caching-system` element.	`String`	Yes.
`advertise`	Advertises this service in the OSGi registry. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.

Example

The following example shows how to use the `wlevs:cache` element in the EPN assembly file. The cache's unique identifier is `cache-id` and its alternate name is `alternative-cache-name`. The caching system to which the cache belongs has an

`id` of `caching-system-id`. The cache has a listener to which the cache sends events; the component that listens to it has an `id` of `tradeListener`.

```
<wlevs:cache id="cache-id" name="alternative-cache-name">
        <wlevs:caching-system ref="caching-system-id"/>
        <wlevs:cache-listener ref="tradeListener" />
</wlevs:cache>
```

2.5 wlevs:cache-listener

Use this element to specify a cache as a source of events to the listening component. The listening component must implement the `com.bea.cache.jcache.CacheListener` interface.

This element is always a child of wlevs:cache.

Attributes

Table 2-4 Attributes of the wlevs:cache-listener Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the component that listens to this cache. Set this attribute to the value of the `id` attribute of the listening component. The listening component can be an adapter or a Spring bean.	`String`	No.

Example

The following example shows how to use the `wlevs:cache-listener` element in the EPN assembly file. The `cache-listener-id` Spring bean listens to events coming from the cache; the class that implements this component, `wlevs.example.MyCacheListener`, must implement the `com.bea.jcache.CacheListener` interface. You must program the `wlevs.example.MyCacheListener` class yourself.

```
<wlevs:caching-system id="caching-system-id"/>
...
<wlevs:cache id="cache-id" name="alternative-cache-name">
    <wlevs:caching-system ref="caching-system-id"/>
    <wlevs:cache-listener ref="cache-listener-id" />
</wlevs:cache>
...
<bean id="cache-listener-id" class="wlevs.example.MyCacheListener"/>
```

2.6 wlevs:cache-loader

spring-wlevs-v12_1_3_0.xsd specifies the Spring bean that implements an object that loads data into a cache.

This element is always a child of wlevs:cache.

Attributes

Table 2-5 Attributes of the wlevs:cache-loader Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the Spring bean that implements the class that loads data into the cache. Set this attribute to the value of the id attribute of the Spring bean. The Spring bean must implement the `com.bea.cache.jcache.CacheLoader` interface.	`String`	Yes.

Example

The following example shows how to use the `wlevs:cache-loader` element in the EPN assembly file. The `cache-loader-id` Spring bean, implemented with the `wlevs.example.MyCacheLoader` class that in turn implements the `com.bea.cache.jcache.CacheLoader` interface, is a bean that loads data into a cache. The cache specifies this loader by pointing to it with the `ref` attribute of the `wlevs:cache-loader` child element.

```
<wlevs:cache id="cache-id" name="alternative-cache-name">
    <wlevs:caching-system ref="caching-system-id"/>
    <wlevs:cache-loader ref="cache-loader-id" />
</wlevs:cache>
...
<bean id="cache-loader-id" class="wlevs.example.MyCacheLoader"/>
```

2.7 wlevs:cache-source

Specifies a cache that supplies data to this processor component. The processor component in turn is associated with an Oracle CQL query that directly references the cache.

Use the `value-type` attribute of the wlevs:cache element to declare the event type of the data supplied by the cache.

This element is a child of only wlevs:processor element.

Attributes

Table 2-6 Attributes of the wlevs:cache-source Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the cache that is a source of data for the processor component. Set this attribute to the value of the `id` attribute of the cache.	`String`	Yes.

Example

The following example shows how to use the `wlevs:cache-source` element in the EPN assembly file. In the example, the processor will have data pushed to it from the

`stream-id` channel as usual; however, the Oracle CQL queries that execute in the processor can also pull data from the `cache-id` cache. When the query processor matches an event type in the FROM clause to an event type supplied by a cache, such as `Company`, the processor pulls instances of that event type from the cache.

```
<wlevs:caching-system id="caching-system-id"/>
  ...
  <wlevs:cache id="cache-id"
               name="alternative-cache-name"
               value-type="Company">
    <wlevs:caching-system ref="caching-system-id"/>
  </wlevs:cache>
  <wlevs:channel id="stream-id"/>
  <wlevs:processor id="processor-id">
    <wlevs:cache-source ref="cache-id">
    <wlevs:source ref="stream-id">
  </wlevs:processor>
```

2.8 wlevs:cache-store

Specifies the Spring bean that implements a custom store that is responsible for writing data from the cache to a backing store, such as a table in a database.

This element is always a child of wlevs:cache.

Attributes

Table 2-7 Attributes of the wlevs:cache-store Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the Spring bean that implements the custom store. Set this attribute to the value of the `id` attribute of the Spring bean. The Spring bean must implement the `com.bea.cache.jcache.CacheStore` interface.	`String`	Yes.

Example

The following example shows how to use the `wlevs:cache-store` element in the EPN assembly file. In the example, the `cache-store-id` Spring bean, implemented with the `wlevs.example.MyCacheStore` class that in turn implements the `com.bea.cache.jcache.CacheStore` interface, is a bean for the custom store, such as a database. The cache specifies this store by pointing to it with the `ref` attribute of the `wlevs:cache-store` child element.

```
  <wlevs:cache id="cache-id" name="alternative-cache-name">
      <wlevs:caching-system ref="caching-system-id"/>
      <wlevs:cache-store ref="cache-store-id" />
  </wlevs:cache>
  ...
 <bean id="cache-store-id" class="wlevs.example.MyCacheStore"/>
```

2.9 wlevs:caching-system

Specifies the caching system used by the application.

Child Elements

The `wlevs:caching-system` application assembly element supports the following child elements:

- wlevs:instance-property
- wlevs:property.

Attributes

Table 2-8 Attributes of the wlevs:caching-system Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Specifies the unique identifier for this caching system. This identifier must correspond to the `<name>` element in the XML configuration file for this caching system	`String`	Yes.
`advertise`	Advertises this service in the OSGi registry. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.
`provider`	Specifies the provider of the caching system if you are using a third-party implementation, such as Oracle Coherence: `<wlevs:caching-system id="myCachingSystem" provider=coherence" />` Typically this attribute corresponds to the `provider-name` attribute of a `<factory>` Spring element that specifies the factory class that creates instances of the third-party caching system. If you do not specify the `provider` or `class` attribute, then the default value is the Oracle Event Processing native caching implementation for local single-JVM caches; this implementation uses an in-memory store.	`String`	No.

Table 2-8 (Cont.) Attributes of the wlevs:caching-system Application Assembly Element

Attribute	Description	Data Type	Required?
`class`	Specifies the Java class that implements this caching system; use this attribute to specify a third-party implementation rather than the Oracle Event Processing native caching implementation. If you specify this attribute, it is assumed that the third-party implementation code resides inside the Oracle Event Processing application bundle itself. The class file to which this attribute points must implement the `com.bea.wlevs.cache.api.CachingSystem` interface. If you do not specify the `provider` or `class` attribute, then the default value is the Oracle Event Processing native caching implementation for local single-JVM caches; this implementation uses an in-memory store.	`String`	No

Example

The following example shows the simplest use of the `wlevs:caching-system` element in the EPN assembly file:

```
<wlevs:caching-system id="caching-system-id"/>
```

The following example shows how to specify a third-party implementation that uses a factory as a provider. In the example, `the.factory.class.name` is a factory for creating some third-party caching system; the provider attribute of `wlevs:caching-system` in turn references it as the caching system implementation for the application.

```
<wlevs:caching-system id ="caching-system-id" provider="caching-provider"/>
<factory id="factory-id" provider-name="caching-provider">
   <class>the.factory.class.name</class>
</factory>
```

2.10 wlevs:channel

Use this element to declare a channel to the Spring application context.

By default, channels assume that events are system time stamped. To configure application time stamped events, see child element wlevs:application-timestamped.

Child Elements

The `wlevs:channel` application assembly element supports the following child elements:

- wlevs:listener
- wlevs:source

- wlevs:instance-property
- wlevs:property
- wlevs:application-timestamped.

Attributes

Table 2-9 Attributes of the wlevs:channel Application Assembly Element

Attribute	Description	Data Type	Required?
`advertise`	Advertises this service in the OSGi registry. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.
`batching`	Specifies whether batching of events should be enabled for the event channel. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.
`event-type`	Specifies the type of events that are allowed to pass through the event channel.	`String`	Yes.
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this channel, if one exists.	`String`	Yes.
`is-relation`	Specifies the kind of events that are allowed to pass through the event channel. Two kind of events are supported: streams and relations. Streams are append-only. Relations support insert, delete, and updates. The default value for this attribute is `false`.	`Boolean`	No.
`listeners`	Specifies the components that listen to this component. Separate multiple components using commas. Set this attribute to the value of the `id` attribute of the element (`wlevs:adapter`, `wlevs:channel`, or `wlevs:processor`) that defines the listening component.	`String`	No.
`max-size`	Specifies the maximum size of the FIFO buffer for this channel as `max-size` number of events. When `max-size = 0`, the channel synchronously passes-through events. When `max-size > 0`, the channel processes events asynchronously, buffering events by the requested size. If `max-threads` is zero, then `max-size` is zero. The default value for this attribute is `1024`.	`integer`	No.

Table 2-9 (Cont.) Attributes of the wlevs:channel Application Assembly Element

Attribute	Description	Data Type	Required?
`max-threads`	Specifies the maximum number of threads that will be used to process events for this channel. When `max-threads = 0`, the channel acts as a pass-through. Event ordering is preserved. When `max-threads > 0`, the channel acts as classic blocking queue, where upstream components are producers of events and the downstream components are the consumers of events. The queue size is defined by the configuration `max-size`. There will be up to `max-threads` number of threads consuming events from the queue. Event ordering is non-deterministic. You can change `max-threads` from 0 to a positive integer (that is, from a pass through to multiple threads) without redeploying. However, if you change `max-threads` from a positive integer to 0 (that is, from multiple threads to a pass through), then you must redeploy your application. If the `max-size` attribute is 0, then setting a value for `max-threads` has no effect. The default value for this attribute is 1.	`integer`	No.
`primary-key`	Specifies the primary key of a relation, as a list of event property names, separated by "," or white-spaces.	`String`	No.
`provider`	Specifies the streaming provider. Valid values are: • `oracle.channel` Default value is `oracle.channel`, which is the out-of-the-box streaming provider.	`String`	No.
`source`	Specifies the component from which the channel sources events. Set this attribute to the value of the `id` attribute of the element (`wlevs:adapter`, `wlevs:channel`, or `wlevs:processor`) that defines the source component.	`String`	No.

Example

The following example shows how to use the `wlevs:channel` element in the EPN assembly file. The example shows how to declare a channel service with unique identifier `fxMarketAmerOut`.

```
<wlevs:channel id="fxMarketAmerOut" />
```

2.11 wlevs:class

Use this element to specify the fully-qualified name of the JavaBean class to use as an event type implementation. This element must be a child of the wlevs:event-type element.

The following example shows how to use the `wlevs:class` element in the EPN assembly file:

```
<wlevs:event-type-repository>
        <wlevs:event-type type-name="SimpleEvent">
            <wlevs:class>com.example.myapp.MyEventType</wlevs:class>
    </wlevs:event-type>
...
</wlevs:event-type-repository>
```

2.12 wlevs:event-bean

Use this element to declare to the Spring application context that an event bean is part of your event processing network (EPN). Event beans are managed by the Oracle Event Processing container, analogous to Spring beans that are managed by the Spring framework. In many ways, event beans and Spring beans are similar so it is up to a developer which one to use in their EPN. Use a Spring bean for legacy integration to Spring. Use an event bean if you want to take full advantage of the additional capabilities of Oracle Event Processing.

For example, you can monitor an event bean using the Oracle Event Processing monitoring framework, make use of the Configuration framework metadata annotations, and record and playback events that pass through the event bean. An event-bean can also participate in the Oracle Event Processing bean life cycle by specifying methods in its EPN assembly file declaration, rather than by implementing Oracle Event Processing API interfaces.

Child Elements

The `wlevs:event-bean` application assembly element supports the following child elements:

- wlevs:listener
- wlevs:instance-property
- wlevs:property

Attributes

Table 2-10 Attributes of the wlevs:event-bean Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this event-bean, if one exists.	`String`	Yes.

Table 2-10 (Cont.) Attributes of the wlevs:event-bean Application Assembly Element

Attribute	Description	Data Type	Required?
`advertise`	Advertises this service in the OSGi registry. Valid values are `true` and `false`. Default is `false`.	`Boolean`	No.
`listeners`	Specifies the components that listen to this component. Set this attribute to the value of the `id` attribute of the element that declared the component.	`String`	No.
`class`	Specifies the Java class that implements this event bean. The bean is not required to implement any Oracle Event Processing interfaces. You must specify either the `provider` or `class` attribute, but not both, otherwise an exception is raised.		
`provider`	Specifies the service provider. In this case, an EDE factory registered with this specific provider name must exist in the application. You must specify either the `provider` or `class` attribute, but not both, otherwise an exception is raised.	`String`	No.
`onevent-method`	Specifies the method of the event bean implementation that corresponds to the life cycle `onEvent` method. Oracle Event Processing invokes this method when the event bean receives an event. By using this life cycle attribute, the event bean implementation does not have to explicitly implement an Oracle Event Processing interface.	`String`	No
`init-method`	Specifies the method of the event bean implementation that corresponds to the life cycle `init` method. Oracle Event Processing invokes this method after it has set all the supplied instance properties. This method allows the bean instance to perform initialization only possible when all bean properties have been set and to throw an exception in the event of misconfiguration. By using this life cycle attribute, the event bean implementation does not have to explicitly implement an Oracle Event Processing interface.	`String`	No

Table 2-10 (Cont.) Attributes of the wlevs:event-bean Application Assembly Element

Attribute	Description	Data Type	Required?
`activate-method`	Specifies the method of the event bean implementation that corresponds to the life cycle `activate` method. Oracle Event Processing invokes this method after the dynamic configuration of the bean has completed. This method allows the bean instance to perform initialization only possible when all dynamic bean properties have been set and the EPN has been wired. By using this life cycle attribute, the event bean implementation does not have to explicitly implement an Oracle Event Processing interface.	`String`	No
`suspend-method`	Specifies the method of the event bean implementation that corresponds to the life cycle `suspend` method. Oracle Event Processing invokes this method when the application is suspended. By using this life cycle attribute, the event bean implementation does not have to explicitly implement an Oracle Event Processing interface.	`String`	No
`destroy-method`	Specifies the method of the event bean implementation that corresponds to the life cycle `destroy` method. Oracle Event Processing invokes this method when the application is stopped. By using this life cycle attribute, the event bean implementation does not have to explicitly implement an Oracle Event Processing interface.	`String`	No

Example

The following example shows how to use the `wlevs:event-bean` element in the EPN assembly file. In the example, the event bean called `myBean` is implemented with the class `com.customer.SomeEventBean`. The component called `myProcessor` receives events from the `myBean` event bean.

```
<wlevs:event-bean id="myBean" class="com.customer.SomeEventBean" >
  <wlevs:listener ref="myProcessor" />
</wlevs:event-bean>
```

2.13 wlevs:event-type

Specifies the definition of an event type used in the Oracle Event Processing application. Once you define the event types of the application, you can reference them in the adapter and business class POJO, as well as the Oracle CQL rules.

You can define an event type in the following ways:

- Create a JavaBean class that represents your event type and specify its fully qualified class name using the `wlevs:class` child element.
- Specify event type properties declaratively with a `wlevs:properties` child element.

You can specify one of either `wlevs:class` or `wlevs:properties` as a child of `wlevs:event-type`, but not both.

The best practice is to define your event type by using the `wlevs:class` child element because you can then reuse the specified JavaBean class, and you control exactly what the event type looks like.

Child Elements

The `wlevs:event-type` application assembly element supports the following child elements:

- wlevs:class
- wlevs:metadata (deprecated)
- wlevs:properties
- wlevs:property.

Attributes

Table 2-11 Attributes of the wlevs:event-type Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Specifies the unique identifier for this event type. If you do not specify this attribute, Oracle Event Processing automatically generates an identifier for you.	`String`	No.
`type-name`	Specifies the name of this event type. This is the name you use whenever you reference the event type in the adapter, business POJO, or Oracle CQL rules.	`String`	Yes.

Table 2-11 (Cont.) Attributes of the wlevs:event-type Application Assembly Element

Attribute	Description	Data Type	Required?
`object-support`	Specifies if Java objects should be fully supported. Allowable values are `true`, `false`, and `object-relational`; default is `object-relational`. If set to `false`, then the Java primitive wrappers (for example, `java.lang.Integer`) and `java.lang.String` are treated as primitive types. If set to `true`, then Java primitive wrappers are treated as classes. If set to `object-relational`, then Java primitive wrappers are treated as relations, rather than as streams.	String	No.

Example

The following example shows how to use the `wlevs:event-type` element in the EPN assembly file. In the example, the name of the event type is `SimpleEvent` and its definition is determined by the `wlevs:property` elements. The values for the type attribute must conform to the `com.bea.wlevs.ede.api.Type` class.

```
<wlevs:event-type-repository>
        <wlevs:event-type id="messagecounts" type-name="SimpleEvent">
            <wlevs:properties>
                <wlevs:property name="msg" type="char" />
                <wlevs:property name="count" type="long" />
                <wlevs:property name="time_stamp" type="timestamp" />
        </wlevs:properties>
    </wlevs:event-type>
...
</wlevs:event-type-repository>
```

2.14 wlevs:event-type-repository

Use this element to group together one or more `wlevs:event-type` elements, each of which is used to register an event type used throughout the application.

Child Element

The `wlevs:event-type-repository` application assembly element supports the wlevs:event-type child element:

Example

The following example shows how to use the `wlevs:event-type-repository` element in the EPN assembly file:

In the example, the `wlevs:event-type-repository` element groups a single `wlevs:event-type` element to declare a single event type: `HelloWorldEvent`. See wlevs:event-type for additional details.

```
<wlevs:event-type-repository>
    <wlevs:event-type type-name="HelloWorldEvent">
        <wlevs:class>
            com.bea.wlevs.event.example.helloworld.HelloWorldEvent
        </wlevs:class>
    </wlevs:event-type>
</wlevs:event-type-repository>
```

2.15 wlevs:expression

Use this element to specify an arithmetic expression in wlevs:application-timestamped to be used as an application time stamp for event processing. For more information, see Application-Timestamped Stream in *Oracle CQL Language Reference for Oracle Event Processing*.

Example

The following example shows how to use `wlevs:expression` element in the EPN assembly file to specify an explicitly application time stamped channel. In the example, the `wlevs:expression` element defines the arithmetic expression used to assign a time stamp to each event.

```
<wlevs:channel id="fxMarketAmerOut" >
    <wlevs:application-timestamped>
        <wlevs:expression>mytime + 10</wlevs:expression>
    </wlevs:application-timestamped>
</wlevs:channel>
```

2.16 wlevs:factory

Use this element to register a factory class as a service. Use of this element decreases the dependency of your application on Spring-OSGi interfaces. The Java source of this factory must implement the `com.bea.wlevs.ede.api.Factory` interface.

The factory element does not allow you to specify service properties. If you need to specify service properties, then you must use the Spring- OSGi `osgi:service` element instead.

Attributes

Table 2-12 Attributes of the wlevs:factory Application Assembly Element

Attribute	Description	Data Type	Required?
`class`	Specifies the Java class that implements the factory. This class must implement the `com.bea.wlevs.ede.api.Factory` interface.	`String`	Yes.
`provider-name`	Specifies the name of this provider. Reference this name later in the component that uses this factory.	`String`	Yes.

Example

The following example shows how to use the `wlevs:factory` element in the EPN assembly file. In the example, the factory implemented by the

`com.customer.MyEventSourceFactory` goes by the provider name of `myEventSourceFactory`.

```
<wlevs:factory provider-name="myEventSourceFactory"
               class="com.customer.MyEventSourceFactory" />
```

2.17 wlevs:function

Use this element to specify a bean that contains user-defined functions for a processor. See *Single-Row Functions* and *Aggregate Functions* in *Oracle CQL Language Reference for Oracle Event Processing*.

This element always has a standard Spring `bean` element either as a child or as a reference that specifies the Spring `bean` that implements the user-defined function.

For a single-row function for an Oracle CQL processor, you can specify one method on the implementing class as the function using the `exec-method` attribute. In this case, the method must be public and must be uniquely identifiable by its name. The method cannot have been overridden. You may define an alias for the `exec-method` name using the `function-name` attribute. In the Oracle CQL query, you may call only the `exec-method` (either by its name or the `function-name` alias).

For an aggregate function on an Oracle CQL processor, the Spring `bean` must implement the following interfaces from the `com.bea.wlevs.processor` package:

- `AggregationFunctionFactory`
- `AggregationFunction`

For an aggregate function, the `exec-method` attribute is not applicable on an Oracle CQL processor.

Attributes

Table 2-13 Attributes of the wlevs:function Application Assembly Element

Attribute	Description	Data Type	Required?
`exec-method`	For a user-defined single-row function on an Oracle CQL processor, this element specifies the method name of the Spring `bean` that implements the function. In this case, the method must be public and must be uniquely identifiable by its name (that is, the method cannot have been overridden). For a user-defined aggregate function on an Oracle CQL processor, this attribute is not applicable.	String	No.

Table 2-13 (Cont.) Attributes of the wlevs:function Application Assembly Element

Attribute	Description	Data Type	Required?
`function-name`	For a user-defined single-row function on an Oracle CQL processor, use this attribute to define an alias for the `exec-method` name. You can then use the `function-name` in your Oracle CQL query instead of the `exec-name`. For a user-defined aggregate function on an Oracle CQL processor, use this attribute to define an alias for the implementing Spring `bean` class name. The default value is the Spring bean name.	`String`	No.
`ref`	Specifies the Spring `bean` that implements the function. Set this attribute to the value of the `id` attribute of the Spring `bean`. This is an alternative to making the Spring `bean` element a child of the `wlevs:function` element.	`String`	No.

The following examples show how to use the `wlevs:function` element and its attributes on both Oracle CQL processors.

Examples

Example 2-1 Implement a single-row, user-defined function for an Oracle CQL processor

```
package com.bea.wlevs.example.function;

public class MyMod {
    public Object execute(int arg0, int arg1) {
        return new Integer(arg0 % arg1);
    }
}
```

Example 2-2 Define a single-row function on an Oracle CQL processor in the assembly file

```
<wlevs:processor id="testProcessor">
    <wlevs:listener ref="providerCache"/>
    <wlevs:listener ref="outputCache"/>
    <wlevs:cache-source ref="testCache"/>
    <wlevs:function function-name="mymod" exec-method="execute" />
        <bean class="com.bea.wlevs.example.function.MyMod"/>
    </wlevs:function>
</wlevs:processor>
```

Example 2-3 Invoke the function in an Oracle CQL query

```
...
<view id="v1" schema="c1 c2 c3 c4"><![CDATA[
    select
        mymod(c1, 100), c2, c3, c4
```

```
    from
        S1
></view>
...
<query id="q1"><![CDATA[
    select * from v1 [partition by c1 rows 1] where c4 - c3 = 2.3
></query>
...
```

Example 2-4 Implement a user-defined aggregate function for an Oracle CQL processor

```
package com.bea.wlevs.test.functions;

import com.bea.wlevs.processor.AggregationFunction;
import com.bea.wlevs.processor.AggregationFunctionFactory;

public class Variance implements AggregationFunctionFactory, AggregationFunction {

    private int count;
    private float sum;
    private float sumSquare;

    public Class<?>[] getArgumentTypes() {
        return new Class<?>[] {Integer.class};
    }

    public Class<?> getReturnType() {
        return Float.class;
    }

    public AggregationFunction newAggregationFunction() {
        return new Variance();
    }

    public void releaseAggregationFunction(AggregationFunction function) {
    }

    public Object handleMinus(Object[] params) {
        if (params != null && params.length == 1) {
            Integer param = (Integer) params[0];
            count--;
            sum -= param;
            sumSquare -= (param * param);
        }

        if (count == 0) {
            return null;
        } else {
            return getVariance();
        }
    }

    public Object handlePlus(Object[] params) {
        if (params != null && params.length == 1) {
            Integer param = (Integer) params[0];
            count++;
            sum += param;
            sumSquare += (param * param);
        }
```

```
        if (count == 0) {
            return null;
        } else {
            return getVariance();
        }
    }

    public Float getVariance() {
        float avg = sum / (float) count;
        float avgSqr = avg * avg;
        float var = sumSquare / (float)count - avgSqr;
        return var;
    }

    public void initialize() {
        count = 0;
        sum = 0.0F;
        sumSquare = 0.0F;
    }

}
```

Example 2-5 Invoke an aggregate function on an Oracle CQL processor in the assembly file.

```
  <wlevs:processor id="testProcessor">
     <wlevs:listener ref="providerCache"/>
     <wlevs:listener ref="outputCache"/>
     <wlevs:cache-source ref="testCache"/>
     <wlevs:function function-name="var">
       <bean class="com.bea.wlevs.test.functions.Variance"/>
     </wlevs:function>
   </wlevs:processor>
```

Example 2-6 Invoke an aggregate function in an Oracle CQL query

```
...
<query id="uda6"><![CDATA[
    select var(c2) from S4[range 3]
></query>
...
```

Example 2-7 Nest a bean element in the assembly file

```
  <wlevs:processor id="testProcessor">
     <wlevs:listener ref="providerCache"/>
     <wlevs:listener ref="outputCache"/>
     <wlevs:cache-source ref="testCache"/>
     <wlevs:function function-name="testfunction">
       <bean class="com.bea.wlevs.example.cache.function.TestFunction"/>
     </wlevs:function>
   </wlevs:processor>
```

Example 2-8 Reference a bean element defined outside of the function in the assembly file

```
 <wlevs:processor id="testProcessor">
     <wlevs:listener ref="providerCache"/>
     <wlevs:listener ref="outputCache"/>
     <wlevs:cache-source ref="testCache"/>
     <wlevs:function function-name="testfunction" ref="testFunctionID" />
   </wlevs:processor>
```

```
  ...
  <bean id="testFunctionID"
class="com.bea.wlevs.example.cache.function.TestFunction"/>
```

2.18 wlevs:instance-property

Specifies the properties that apply to the create stage instance of the component to which this is a child element. This allows declarative configuration of user-defined stage properties.

For example, when you specify a `wlevs:instance-property` for a `wlevs:event-bean`, Oracle Event Processing looks for a corresponding setter method on the Java class you implement, not on the `com.bea.wlevs.spring.EventBeanFactoryBean` that instantiates your class. To specify a property on the factory, use wlevs:property

This element is used only as a child of wlevs:adapter, wlevs:event-bean, wlevs:processor, wlevs:channel, or wlevs:caching-system.

The `wlevs:instance-property` element is defined as the Spring `propertyType` type. For more information about the Spring data type, see `http://www.springframework.org/schema/beans/spring-beans-2.0.xsd`.

Child Elements

You can specify one of the following standard Spring elements as a child element of the `wlevs:instance-property` element:

- `meta`
- `bean`
- `ref`
- `idref`
- `value`
- `null`
- `list`
- `set`
- `map`
- `props`.

Attributes

Table 2-14 Attributes of the wlevs:instance-property Application Assembly Element

Attribute	Description	Data Type	Required?
`name`	Specifies the name of the property, following JavaBean naming conventions.	`String`	Yes.

Table 2-14 (Cont.) Attributes of the wlevs:instance-property Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	A short-cut alternative to a nested `<ref bean='...'/>` element.	`String`	No.
`value`	A short-cut alternative to a nested `<value>...</value>` element.	`String`	No.

Examples

The following example shows how to use the `wlevs:instance-property` element in the EPN assembly file:

```
<wlevs:event-bean id="pubsubCounterBeanRemote"
    class="com.oracle.cep.example.httppubsub.RemoteEventCounter">
    <wlevs:listener ref="pubsubRemote" />
    <wlevs:instance-property name="expectedEvents" value="4000" />
</wlevs:event-bean>
```

In the example, the event bean `com.oracle.cep.example.httppubsub.RemoteEventCounter` class provides an appropriate setter method:

```
    private int expectedEvents;

    public void setExpectedEvents(String expectedEvents) {
        this.expectedEvents = new Integer(expectedEvents).intValue();
    }
```

Note that the `instance-property` is of type `String`. Your setter method must convert this if necessary. In this example, the `String` is converted to an `int` value.

The name of the setter method must conform to JavaBean naming conventions. In this example, the setter name is `setExpectedEvents` and this corresponds to the `wlevs:instance-property` element `name` attribute value `expectedEvents`, according to JavaBean conventions. If the `name` attribute value is `obj` and the setter method name is `setObject`, Oracle Event Processing will throw an Invalid Property exception. In this case, the setter name should be `setObj`.

2.19 wlevs:listener

Specifies the component that listens to the component to which this element is a child. A listener can be an instance of any other component. You can also nest the definition of a component within a particular `wlevs:listener` component to specify the component that listens to the parent.

Caution:

Nested definitions are not eligible for dynamic configuration or monitoring.

This element is always a child of wlevs:adapter, wlevs:processor, wlevs:channel, or wlevs:caching-system.

Attributes

Table 2-15 Attributes of the `wlevs:listener` Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the component that listens to the parent component. Set this attribute to the value of the `id` attribute of the listener component. You do not specify this attribute if you are nesting listeners.	`String`	No.

Example

The following example shows how to use the `wlevs:listener` element in the EPN assembly file. In the example, the `hellworldOutstream` component listens to the `helloworldProcessor` component. It is assumed that the EPN assembly file also contains a declaration for a `wlevs:adapter`, `wlevs:channel`, or `wlevs:processor` element whose unique identifier is `helloworldOutstream`.

```
<wlevs:processor id="helloworldProcessor">
    <wlevs:listener ref="helloworldOutstream"/>
</wlevs:processor>
```

2.20 wlevs:metadata

Specifies the definition of an event type by listing its fields as a group of Spring `entry` elements. When you define an event type this way, Oracle Event Processing automatically generates the Java class for you.

Use the `key` attribute of the `entry` element to specify the name of a field and the `value` attribute to specify the Java class that represents the field's data type.

This element is used only as a child of wlevs:event-type.

The `wlevs:metadata` element is defined as the Spring `mapType` type; for additional details of this Spring data type, see the `http://www.springframework.org/schema/beans/spring-beans-2.0.xsd`.

Child Elements

The `wlevs:metadata` element can have one or more standard Spring `entry` child elements as defined in the `http://www.springframework.org/schema/beans/spring-beans-2.0.xsd`.

Attributes

Table 2-16 Attributes of the wlevs:metadata Application Assembly Element

Attribute	Description	Data Type	Required?
`key-type`	The default fully qualified class name of a Java data type for nested `entry` elements. You use this attribute *only* if you have nested `entry` elements.	`String`	No.

Example

The following example shows how to use the `wlevs:metadata` element in the EPN assembly file. In the example, the `wlevs:metadata` element groups together four standard Spring `entry` elements that represent the four fields of the `ForeignExchangeEvent`: `symbol`, `price`, `fromRate`, and `toRate`. The data types of the fields are `java.lang.String`, `java.lang.Double`, `java.lang.String`, and `java.lang.String`, respectively.

```
<wlevs:event-type type-name="ForeignExchangeEvent">
    <wlevs:metadata>
        <entry key="symbol" value="java.lang.String"/>
        <entry key="price" value="java.lang.Double"/>
        <entry key="fromRate" value="java.lang.String"/>
        <entry key="toRate" value="java.lang.String"/>
    </wlevs:metadata>
    ...
</wlevs:event-type>
```

2.21 wlevs:processor

Use this element to declare a processor to the Spring application context.

Child Elements

The `wlevs:processor` Spring element supports the following child elements:

- wlevs:instance-property
- wlevs:listener
- wlevs:property
- wlevs:cache-source
- wlevs:table-source
- wlevs:function.

Attributes

Table 2-17 Attributes of the wlevs:processor Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this processor; this is how Oracle Event Processing knows which Oracle CQL rules to execute for which processor component in your network.	`String`	Yes.
`advertise`	Advertises this service in the OSGi registry. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.

Table 2-17 (Cont.) Attributes of the wlevs:processor Application Assembly Element

Attribute	Description	Data Type	Required?
`listeners`	Specifies the components that listen to this component. Set this attribute to the value of the `id` attribute of the element that declared the component.	`String`	No.
`provider`	Specifies the language provider of the processor, such as Oracle CQL. Valid value is cql. The default value is `cql`.	`String`	No.
`queryURL`	Specifies a URL that points to an Oracle CQL rules definition file for this processor.	`String`.	No.

Example

The following example shows how to use the `wlevs:processor` element in the EPN assembly file. The example shows how to declare a processor with ID `spreader`. This means that in the processor configuration file that contains the Oracle CQL rules for this processor, the `name` element must contain the value `spreader`. This way Oracle Event Processing knows which Oracle CQL rules it must file for this particular processor.

```
<wlevs:processor id="spreader" />
```

2.22 wlevs:properties

Defines the list of properties for an event type. Use this element when you are defining an event type declaratively, such as for a type based on a tuple or `java.util.Map`. For an event type created from a JavaBean class, allow properties to be defined by accessor methods in the class.

Child Elements

The `wlevs:properties` application assembly element supports the wlevs:property child elements:

Attributes

Table 2-18 Attributes of the wlevs:properties Application Assembly Element

Attribute	Description	Data Type	Required?
`type`	Specifies the type that this event type should be created from. Allowable values are tuple and map; default is tuple. If this attribute's value is `map`, then Oracle Event Processing will instantiate the event type as a `java.util.Map` instance. Otherwise, the type will be instantiated as a tuple.	`String`	No.

Example

The following example shows how to use the `wlevs:properties` element in the EPN assembly file. In the example, the name of the event type is `SimpleEvent` and its definition is determined by the `wlevs:property` elements. The values for the type attribute must conform to the `com.bea.wlevs.ede.api.Type` class.

```
<wlevs:event-type-repository>
        <wlevs:event-type type-name="SimpleEvent">
            <wlevs:properties>
                <wlevs:property name="msg" type="char" />
                <wlevs:property name="count" type="long" />
                <wlevs:property name="time_stamp" type="timestamp" />
        </wlevs:properties>
    </wlevs:event-type>
...
</wlevs:event-type-repository>
```

2.23 wlevs:property

Defines the property of an event type that you create declaratively, such as an event type based on a tuple or `java.util.Map`. You use this `wlevs:property` element as a child of the wlevs:properties element.

Note that this element is different from the wlevs:property element that is an extension of the Spring beans property element. This element must always be used as a child of the `wlevs:properties` element.

Attributes

Table 2-19 Attributes of the wlevs:property Application Assembly Element

Attribute	Description	Data Type	Required ?
`name`	Name of the event property.	`String`	Yes.
`type`	Type of the event property. If this event type is defined as a tuple, then the type attribute value must be one of the OCEP native types defined by com.bea.wlevs.ede.api.Type. Those include `bigint`, `boolean`, `byte`, `char`, `double`, `float`, `int`, `interval`, `object`, `sqlxml`, `timestamp`, `unknown`, and `xmltype`. If this event type is defined as a map, then the type attribute value is the fully qualified name of a Java class that must be available in the class loader of the application that defines the event type. The string used as the Java class name must conform to the same rules as Class.forName(). In addition, Java primitives can be used (such as `int` and `float`). Finally, you can specify an array by appending the characters '[]' to the Java class name	`String`	Yes.
`length`	If the property is of array type, this specifies the length of the array. If the type is not an array and length is specified, it will be ignored. .	`String`	No.

Example

The following example shows how to use the `wlevs:property` element in the EPN assembly file. In the example, the name of the event type is `SimpleEvent` and its definition is determined by the `wlevs:property` elements. The values for the type attribute must conform to the `com.bea.wlevs.ede.api.Type` class.

```
<wlevs:event-type-repository>
        <wlevs:event-type type-name="SimpleEvent">
            <wlevs:properties>
                <wlevs:property name="msg" type="char" />
                <wlevs:property name="count" type="long" />
                <wlevs:property name="time_stamp" type="timestamp" />
        </wlevs:properties>
    </wlevs:event-type>
...
</wlevs:event-type-repository>
```

2.24 wlevs:property

Specifies a custom property to apply to the event type.

For example, when you specify a `wlevs:property` for a `wlevs:event-bean`, Oracle Event Processing looks for a corresponding setter method on the `com.bea.wlevs.spring.EventBeanFactoryBean` that instantiates your Java class, not on the Java class you implement. To specify a property on your Java class, use wlevs:instance-property.

This element is used only as a child of wlevs:adapter, wlevs:event-bean, wlevs:event-type, wlevs:processor, wlevs:channel, or wlevs:caching-system.

The `wlevs:property` element is defined as the Spring `propertyType` type. For more information about the Spring data type, see `http://www.springframework.org/schema/beans/spring-beans-2.0.xsd`

Child Elements

You can specify one of the following standard Spring elements as a child element of the `wlevs:property` element:

- `meta`
- `bean`
- `ref`
- `idref`
- `value`
- `null`
- `list`
- `set`
- `map`
- `props`.

Attributes

Table 2-20 Attributes of the wlevs:property Application Assembly Element

Attribute	Description	Data Type	Required?
`name`	Specifies the name of the property, following JavaBean naming conventions.	`String`	Yes.
`ref`	A short-cut alternative to a nested `<ref bean='...'/>` element.	`String`	No.
`value`	A short-cut alternative to a nested `<value>...</value>` element.	`String`	No.

Example

The following example shows how to use the `wlevs:property` element in the EPN assembly file. In the example, the `wlevs:property` element defines a custom property of the `ForeignExchangeEvent` called `builderFactory`. The property uses the standard Spring `bean` element to specify the Spring bean used as a factory to create `ForeignExchangeEvents`.

```
<wlevs:event-type type-name="ForeignExchangeEvent">
   <wlevs:metadata>
      <entry key="symbol" value="java.lang.String"/>
      <entry key="price" value="java.lang.Double"/>
   </wlevs:metadata>
   <wlevs:property name="builderFactory">
     <bean id="builderFactory"
           class="com.bea.wlevs.example.fx.ForeignExchangeBuilderFactory"/>
   </wlevs:property>
</wlevs:event-type>
```

2.25 wlevs:source

Specifies an event source for this component. The event source is the component from which the events are coming. Specifying an event source is equivalent to specifying this component as an event listener to another component.

You can also nest the definition of a component within a particular `wlevs:source` component to specify the component source. This element is a child of wlevs:channel or wlevs:processor.

Caution:

Nested definitions are not eligible for dynamic configuration or monitoring.

Attributes

Table 2-21 Attributes of the wlevs:source Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the source of the channel to which this element is a child. Set this attribute to the value of the `id` attribute of the source component. You do not specify this attribute if you are nesting sources.	`String`	No.

Example

The following example shows how to use the `wlevs:source` element in the EPN assembly file. In the example, the component with `id helloworldAdapter` is the source of the `helloworldInstream` channel component.

```
<wlevs:channel id="helloworldInstream">
    <wlevs:listener ref="helloworldProcessor"/>
    <wlevs:source ref="helloworldAdapter"/>
</wlevs:channel>
```

2.26 wlevs:table

Specifies a relational database table that can work as an event sink or an event source.

2.26.1 Table Source

A relational database table that supplies event data to one or more processor components. You associate the processor components with an Oracle CQL query that directly references the table source.

Attributes

Table 2-22 Attributes of the wlevs:table Application Assembly Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this table.	`String`	Yes.
`event-type`	The name of the event type associated with this table as defined in the event type repository.	`String`	Yes.
`data-source`	The name of the relational data source defined in the Oracle Event Processing server configuration file used to access this database table.	`String`	Yes.
table-name	The name of the relational database table. When you use `wlevs:table` as a table source, the `table-name` is optional. When you use `wlevs:table` as a table sink, the `table-name` is required.	String	No

Table 2-22 (Cont.) Attributes of the wlevs:table Application Assembly Element

Attribute	Description	Data Type	Required ?
external-rows-threshold	Specify the external rows threshold when you use a table source as an external source for joining. Does not apply to table sinks.	Long	No

Example

The following example shows how to use the `wlevs:table` element in the EPN assembly file. In this example, a wlevs:processor references the table using its wlevs:table-source element.

```
<wlevs:table id="Stock" event-type="StockEvent" data-source="StockDs" />

<wlevs:processor id="proc">
    <wlevs:table-source ref="Stock" />
</wlevs:processor>
```

2.26.2 Table Sink

A relational database table component that is a table tag in an Oracle CQL processor that receives event data coming into the EPN and stores that data in a persistent data store. This feature is more flexible than the recording feature because you can control the structure of the relational database table. In contrast, the recording feature stores events in a format that works best for the record and playback feature.

After events are stored in the persistent data source, they are sent to the next stage in the EPN. When you create the persistent database table, make sure the column names are the same as the event type property names.

Attributes

Table 2-23 Attributes of the wlevs:table Application Assembly Element

Attribute	Description	Data Type	Required ?
`id`	Unique identifier for this component. This identifier must correspond to the `<name>` element in the XML configuration file for this table.	`String`	Yes
`event-type`	The name of the event type associated with this table as defined in the event type repository.	`String`	Yes
`data-source`	The name of the relational data source defined in the Oracle Event Processing server configuration file used to access this database table.	`String`	`Yes`
table-name	The name of the relational database table. When you use `wlevs:table` as a table source, the `table-name` is optional. When you use `wlevs:table` as a table sink, the `table-name` is required.	String	Yes

Table 2-23 (Cont.) Attributes of the wlevs:table Application Assembly Element

Attribute	Description	Data Type	Required ?
key-properties	Forms the unique key value for the relational database table. The unique key is required for update and delete operations executed by an event sink.	String	Yes

Example

The following example is an added final stage in the HelloWorld example that is a table sink. The table sink stores events of type `HelloWorldEvent` to a data source named `helloDataSource`.

```
<wlevs:channel id="helloworldOutputChannel" event-type="HelloWorldEvent"
               advertise="true">
  <wlevs:listener ref="tableSink"/>
  <wlevs:source ref="helloworldProcessor"/>
</wlevs:channel>
<wlevs:table id="tableSink" event-type="HelloWorldEvent" key-properties="message"
             data-source="helloDataSource" table-name="HelloMessages"/>
```

2.27 wlevs:table-source

Specifies a relational database table that supplies data to this processor component. The processor component in turn is associated with an Oracle CQL query that directly references the table. This element is a child of only wlevs:processor element.

Attributes

Table 2-24 Attributes of the wlevs:table-source Application Assembly Element

Attribute	Description	Data Type	Required?
`ref`	Specifies the relational database table that is a source of data for the processor component. Set `ref` to the `id` attribute value of a wlevs:table element.	`String`	Yes.

Example

The following example shows how to use the `wlevs:table-source` element in the assembly file:

```
<wlevs:table id="Stock" event-type="StockEvent" data-source="StockDs" />

<wlevs:processor id="proc">
    <wlevs:table-source ref="Stock" />
</wlevs:processor>
```

3

Component Element Hierarchies

The component element hierarchies show the inheritance relationships for the elements used to configure EPN components. The component elements are defined in the `Oracle/Middleware/my_oep/oep/xsd/wlevs_application_config.xsd` schema and described in Component Configuration Schema and Event Record and Playback Schema.

This chapter presents the hierarchies for the top-level component elements. All of the elements mentioned in this chapter link to descriptions that are in the remaining chapters of this guide.

This chapter includes the following sections, which is also a list of the top-level elements:

- adapter
- http-pub-sub-adapter
- jms-adapter
- edn-adapter
- csv-adapter
- obr-adapter
- channel
- processor
- event-bean
- caching-system
- coherence-caching-system
- diagnostic-profiles
- jaxb-mapper
- rest-adapter
- csv-mapper
- rmi-adapter.

3.1 adapter

All Oracle Event Processing adapters have the adapter element and all of its child elements.

- **adapter**
  - name
  - record-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range
      - start
      - end
    - time-range-offset
      - start
      - duration
    - batch-size
    - batch-time-out
  - playback-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range
      - start
      - end
    - time-range-offset
      - start
      - duration
    - schedule-time-range-offset
      - start
      - duration
  - work-manager-name
  - netio
    - provider-name
    - num-threads
    - accept-backlog

3.2 http-pub-sub-adapter

- **http-pub-sub-adapter**
  - name
  - record-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters

- parameter
  - name
  - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- batch-size
- batch-time-out

playback-parameters
- dataset-name
- event-type-list
  - event-type
- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- schedule-time-range
  - start
  - end
- schedule-time-range-offset
  - start
  - duration

work-manager-name
netio
- netio
- num-threads
- accept-backlog

`One of:`
- server-context-path
- server-url

channel (http-pub-sub-adapter Child Element)
event-type
user
`One of:`
- password
- password

3.3 jms-adapter

jms-adapter
- name
- record-parameters
  - dataset-name

- event-type-list
  - event-type
- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- batch-size
- batch-time-out

playback-parameters

- dataset-name
- event-type-list
  - event-type
- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- schedule-time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- event-type

jndi-provider-url

jndi-factory

connection-jndi-name

`One of:`

- destination-jndi-name
- destination-name

user

`One of:`

- password
- encrypted-password

connection-user

`One of:`

- connection-password
- connection-encrypted-password

work-manager

concurrent-consumers

message-selector

- session-ack-mode-name
- session-transacted
- delivery-mode
- bindings (jms-adapter)
  - group-binding
    - param
- destination-type
- durable-subscription
- durable-subscription-name

3.4 edn-adapter

- **edn-adapter**
  - name
  - record-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range
      - start
      - end
    - time-range-offset
      - start
      - duration
    - batch-size
    - batch-time-out
  - playback-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range
      - start
      - end
    - time-range-offset
      - start
      - duration
    - schedule-time-range
      - start
      - end
    - schedule-time-range-offset
      - start
      - duration
  - work-manager-name

- netio
  - provider-name
  - num-threads
  - accept-backlog
- edl-file
- schema-file
- validate
- package-name
- metadata
- raw-xml-content
- jndi-provider-url
- jndi-factory
- user
- one of:
  - password
  - encrypted-password
- delivery-mode
- priority
- time-to-live
- durable-subscription-name
- client-id

3.5 csv-adapter

- **csv-adapter**
  - name
  - record-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range
      - start
      - end
    - time-range-offset
      - start
      - duration
    - batch-size
    - batch-time-out
  - playback-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range

- start
- end
- time-range-offset
  - start
  - duration
- time-range-offset
  - start
  - end
- schedule-time-range-offset
  - start
  - duration
- work-manager-name
- netio
  - provider-name
  - num-threads
  - accept-backlog
- event-interval
  - unit
- source-url
- initial-delay
  - unit
- output-file
- append

3.6 obr-adapter

- **obr-adapter**
  - name
  - record-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size
    - max-threads
    - time-range
      - start
      - end
    - time-range-offset
      - start
      - duration
    - batch-size
    - batch-time-out
  - playback-parameters
    - dataset-name
    - event-type-list
      - event-type
    - provider-name
    - store-policy-parameters
      - parameter
        - name
        - value
    - max-size

- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- schedule-time-range
  - start
  - end
- schedule-time-range-offset
  - start
  - duration

work-manager-name
netio
- provider-name
- num-threads
- accept-backlog

dictionary-url
decision-function

3.7 channel

channel
- name
- record-parameters
  - dataset-name
  - event-type-list
    - event-type-name
  - provider-name
  - store-policy-parameters
    - parameter
      - name
      - value
  - max-size
  - max-threads
  - recording-session-name

`One of:`
- time-range
  - start
  - end

`or`
- time-range-offset
  - start
  - duration
- batch-size
- batch-time-out

playback-parameters
- playback-parameters
- event-type-list
  - event-type
- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size

- max-threads

One of:

- time-range
  - start
  - end

or

- time-range-offset
  - start
  - duration

or

- schedule-time-range
  - start
  - end

or

- schedule-time-range-offset
  - start
  - duration
- playback-speed
- repeat

- inject-parameters
  - active
  - channel-name
    - trace-parameters
      - active
      - channel-name
    - max-size
    - max-threads
    - selector
    - heartbeat
    - partition-order-capacity
    - offer-timeout
    - fail-when-rejected

3.8 processor

processor (Oracle CQL)

- name
- record-parameters
  - dataset-name
  - event-type-list
    - event-type
  - provider-name
  - store-policy-parameters
    - parameter
      - name
      - value
  - max-size
  - max-threads
  - time-range
    - start
    - end
  - time-range-offset
    - start
    - duration
  - batch-size
  - batch-time-out
- playback-parameters

- dataset-name
- event-type-list
  - event-type
- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- schedule-time-range
  - start
  - end
- time-range-offset
  - start
  - duration

rules
- query
- view

bindings (processor)
- params

3.9 event-bean

event-bean
- name
- record-parameters
  - dataset-name
  - event-type-list
    - event-type
  - provider-name
  - store-policy-parameters
    - parameter
      - name
      - value
  - max-size
  - max-threads
  - time-range
    - start
    - end
  - time-range-offset
    - start
    - duration
  - batch-size
  - batch-time-out
- playback-parameters
  - dataset-name
  - event-type-list
    - event-type
  - provider-name
  - store-policy-parameters

- parameter
  - name
  - value
- max-threads
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- schedule-time-range
  - start
  - end
- schedule-time-range-offset
  - start
  - duration

3.10 caching-system

- **caching-system**
  - name
  - cache
    - name
    - max-size
    - eviction-policy
    - time-to-live
    - idle-time
    - `One of:`
      - write-none
      - write-through
      - write-behind
        - work-manager-name
          - batch-size
          - buffer-size
          - buffer-write-attempts
          - buffer-write-timeout
    - work-manager-name
    - listeners

3.11 coherence-caching-system

- **coherence-caching-system**
  - name
  - coherence-cache-config
  - coherence-cluster-config

3.12 diagnostic-profiles

- **diagnostic-profiles**
  - name
  - profile
    - name
    - enabled
    - start-stage
    - max-latency

- name
- collect-interval
  - amount
  - unit
- start-location
  - application
  - stage
  - direction
- end-location
  - application
  - stage
  - direction

average-latency
- name
- collect-interval
  - amount
  - unit
- start-location
  - application
  - stage
  - direction
- end-location
  - application
  - stage
  - direction
- threshold

throughput
- name
- throughput-interval
  - amount
  - unit
- average-interval
  - amount
  - unit
- location
  - application
  - stage
  - direction

3.13 jaxb-mapper

jaxb-mapper
- name
- event-type-name
- context-path
- metadata
- validate
- schema-file

3.14 rest-adapter

rest-adapter
- name
- record-parameters
  - dataset-name
  - event-type-list
    - event-type

- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- batch-size
- batch-time-out

playback-parameters
- dataset-name
- event-type-list
  - event-type
- provider-name
- store-policy-parameters
  - parameter
    - name
    - value
- max-size
- max-threads
- time-range
  - start
  - end
- time-range-offset
  - start
  - duration
- schedule-time-range
  - start
  - end
- schedule-time-range-offset
  - start
  - duration

work-manager-name

netio
- provider-name
- num-threads
- accept-backlog

event-type-name

context-path

3.15 csv-mapper

csv-mapper
- name
- event-type-name
- context-path
- metadata
- validate
- schema
- event-interval
  - unit

initial-delay
unit

3.16 rmi-adapter

rmi-adapter
name
record-parameters
dataset-name
event-type-list
event-type
provider-name
store-policy-parameters
parameter
name
value
max-size
max-threads
time-range
start
end
time-range-offset
start
duration
batch-size
batch-time-out
playback-parameters
dataset-name
event-type-list
event-type
provider-name
store-policy-parameters
parameter
name
value
max-size
max-threads
time-range
start
end
time-range-offset
start
duration
schedule-time-range
start
end
schedule-time-range-offset
start
duration
work-manager-name
netio
provider-name
num-threads
accept-backlog
jndi-name
jndi-provider-url
jndi-factory

user
one of:
  password
  encrypted-password

4

Component Configuration Schema

This chapter provides a reference to the elements of the `Oracle/Middleware/my_osa/osa/wlevs_application_config.xsd` schema. This schema is behind the XML files you use to configure Oracle Event Processing application components such as adapters, channels, caching systems, and event beans.

This chapter includes the following sections:

- collect-interval
- concurrent-consumers
- configuration
- config-name
- config-value
- connection-jndi-name
- connection-encrypted-password
- connection-password
- connection-user
- context-path
- csv-adapter
- csv-mapper
- database
- decision-function
- default-session
- delivery-mode
- destination-jndi-name
- destination-name
- destination-type
- diagnostic-profiles
- dictionary-url
- direction
- durable-subscription
- durable-subscription-name
- duration
- edl-file
- edn-adapter
- enabled
- encrypted-password
- end
- end-location
- event-bean

- event-interval
- event-type
- event-type-name
- eviction-policy
- fail-when-rejected
- group-binding
- heartbeat
- http-pub-sub-adapter
- idle-time
- initial-delay
- inject-parameters
- jaxb-mapper
- jms-adapter
- jndi-factory
- jndi-name
- jndi-provider-url
- listeners
- location
- max-latency
- max-size
- max-threads
- message-selector
- metadata
- name
- netio
- num-threads
- obr-adapter
- offer-timeout
- output-file
- package-name
- param
- parameter

- params
- partition-order-capacity
- password
- playback-parameters
- priority
- processor (Oracle CQL)
- profile
- query
- raw-xml-content
- record-parameters
- repeat
- rest-adapter
- rest-outbound-adapter
- rmi-adapter
- rules
- schema
- schema-file
- selector
- server-context-path
- server-url
- session
- session-ack-mode-name
- session-transacted
- source-url
- stage
- start
- start-location
- start-stage
- threshold
- throughput
- throughput-interval
- time-to-live

- trace-parameters
- unit
- user
- validate
- value
- view
- work-manager
- work-manager-name
- write-behind
- write-none
- write-through.

4.1 accept-backlog

Use this element to define the maximum number of pending connections allowed on a socket. This element is only applicable in a netio element. This element has no child elements and no attributes.

The following example shows how to use the `accept-backlog` element in the component configuration file:

```
<netio> <provider-name>providerCache</provider-name>
        <num-threads>1000</num-threads>
        <accept-backlog>50</accept-backlog>
</netio>
```

4.2 active

Use the active element to turn event tracing on and off. When `true`, event tracing is on. When `false`, event tracing is off. When the `active` element value is set to `false`, the channel-name value is ignored. This element has no child elements and no attributes.

The following example shows how to configure a processor for event tracing. The trace-parameters element's active child element specifies that tracing is on, while the channel-name element specifies the HTTP pub-sub channel to which traced elements should be sent.

```
<processor>
    <name>FindCrossRates</name>
    <trace-parameters>
        <active>true</active>
        <channel-name>/NonClusteredServer/fx/FindCrossRates/output</channel-name>
    </trace-parameters>
    <rules>
        <!-- Query rules omitted. -->
    </rules>
</processor>
```

4.3 adapter

Use this element to define a custom adapter component. This element has the following child elements and no attributes.

- name
- record-parameters
- playback-parameters
- work-manager-name
- netio

See also the following specialized adapter elements: http-pub-sub-adapter, jms-adapter, edn-adapter, csv-adapter, obr-adapter, or rmi-adapter elements.

The following example shows how to use the `adapter` element in the component configuration file. In the example, the adapter's unique identifier is `trackdata`.

```
<adapter>
    <name>trackdata</name>
    <symbols>
        <symbol>BEAS</symbol>
        <symbol>IBM</symbol>
    </symbols>
 </adapter>
```

The `adapter` component configuration element supports the following child elements and has no attributes:

4.4 amount

Use this element to define the time duration for a diagnostic profile. The `amount` element has no child elements and no attributes.

This element can be used with any of the following elements:

- average-interval
- collect-interval
- throughput-interval

The following example shows how to use the `amount` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
```

```
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.5 append

Use the `append` element with a CSV outbound adapter to specify whether event data should be appended to the output CSV file if it exists. When `true`, Oracle Event Processing appends data to an existing CSV output file. When `false`, Oracle Event Processing creates a new CSV file or overwrites an existing CSV file of the same name. This element has no child elements and no attributes.

The following example shows the assembly file entry for the `StockTradeCSVOutboundAdapter`. The `append` element value is `false`.

```
<wlevs:adapter id="StockTradeCSVOutboundAdapter" provider="csv-outbound">
    <wlevs:instance-property name="eventType" value="TradeEvent"/>
    <wlevs:instance-property name="outputFile"
        value="/scratch/mpawlan/oep9-19/oep/utils/load-generator/StockData.csv"/>
    <wlevs:instance-property name="append" value="false"/>
</wlevs:adapter>
```

4.6 application

Use this element to define the type of application Oracle Event Processing server applies to a foreign stage. In a diagnostic profile, this element always has a value of `diagnostic`. This element has no child elements and no attributes.

The following example shows how to use the `application` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.7 average-interval

Use the `average-interval` element to define the time interval for which you want to gather metrics. This element has the following child elements and no attributes.

- amount
- unit.

The following example shows how to use the `average-interval` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <throughput>
            <throughput-interval>
                <amount>100000</amount>
                <unit>MICROSECONDS</unit>
            </throughput-interval>
            <average-interval>
                <amount>100000000</amount>
                <unit>NANOSECONDS</unit>
            </average-interval>
            <location>
                <application>diagnostic</application>
                <stage>AlertEventStream</stage>
                <direction>INBOUND</direction>
            </location>
        </throughput>
    </profile>
</diagnostic-profiles>
```

4.8 average-latency

Use this element to define an average latency calculation in a diagnostic profile.

The following example shows how to use the `average-latency` element in the component configuration file:

- name
- collect-interval
- start-location
- end-location
- threshold

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <average-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
```

```
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
            <threshhold>
                <amount>100</amount>
                <unit>MILLISECONDS</unit>
            </threshhold>
        </average-latency>
    </profile>
</diagnostic-profiles>
```

4.9 bindings (jms-adapter)

In an Oracle Event Processing application, you can use the `com.oracle.cep.cluster.hagroups.ActiveActiveGroupBean` class to partition an incoming JMS stream with notification groups. Then, use the `jms-adapter` `bindings` element to associate a notification group with a particular `message-selector` value.

This element has the group-binding child element and no attributes.

The following example shows how to use the `bindings` element in the component configuration file. When an application with this configuration deploys to a server and the server has a `cluster` element `groups` child element that contains `ActiveActiveGroupBean_group1`, the following happens:

- The application processes events with an `acctid` property that is greater than 400.
- The `CONDITION` parameter is defined as `acctid > 400`.

```
<jms-adapter>
    <name>JMSInboundAdapter</name>
    <event-type>StockTick</event-type>
    <jndi-provider-url>t3://ppurich-pc:7001</jndi-provider-url>
    <destination-jndi-name>./Topic1</destination-jndi-name>
    <user>weblogic</user>
    <password>weblogic1</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>true</session-transacted>
    <message-selector>${CONDITION}</message-selector>
    <bindings>
        <group-binding group-id="ActiveActiveGroupBean_group1">
            <param id="CONDITION">acctid > 400</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group2">
            <param id="CONDITION">acctid BETWEEN 301 AND 400</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group3">
            <param id="CONDITION">acctid BETWEEN 201 AND 300</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group4">
            <param id="CONDITION">acctid <= 200</param>
        </group-binding>
     </bindings>
</jms-adapter>
```

4.10 bindings (processor)

Use the processor `bindings` element to define bindings for one or more parameterized Oracle CQL rules in a processor component. The `bindings` component configuration element has the `binding` child element and no attributes.

The following example shows how to use the `bindings` element in the component configuration file:

```
<processor>
    <name>processor1</name>
        <record-parameters>
            <dataset-name>test1data</dataset-name>
            <event-type-list>
                <event-type>SimpleEvent</event-type>
            </event-type-list>
            <provider-name>test-rdbms-provider</provider-name>
            <batch-size>1</batch-size>
            <batch-time-out>10</batch-time-out>
        </record-parameters>
        <rules>
            <rule id="rule1"><![CDATA[
                select stockSymbol, avg(price) as percentage
                from StockTick retain 5 events
                where stockSymbol=?
                having avg(price)  > ? or avg(price)  < ?
            ></rule>
        </rules>
        <bindings>
            <binding id="rule1">
                <params>BEAS,10.0,-10.0</params>
                <params id="IBM">IBM,5.0,5.0</params>
            </binding>
        </bindings>
</processor>
```

4.11 buffer-size

Use the `buffer-size` element to define the size of the internal store buffer that's used to temporarily hold asynchronous updates that need to be written to the store. Does not support dynamic updates. This element has no child elements and no attributes.

The following example shows how to use the `buffer-size` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-behind>
            <work-manager-name>JettyWorkManager</work-manager-name>
            <batch-size>100</batch-size>
            <buffer-size>100</buffer-size>
            <buffer-write-attempts>100</buffer-write-attempts>
            <buffer-write-timeout>100</buffer-write-timeout>
        </write-behind>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.12 buffer-write-attempts

Use the `buffer-write-attempts` element to define the number of attempts that the user thread will make to write to the store buffer. The user thread is the thread that

creates or updates a cache entry. If the user thread cannot write to the store buffer (all write attempts fail), it will invoke the store synchronously. This element may be changed dynamically.

This element has no child elements and no attributes.

The following example shows how to use the `buffer-write-attempts` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-behind>
            <work-manager-name>JettyWorkManager</work-manager-name>
            <batch-size>100</batch-size>
            <buffer-size>100</buffer-size>
            <buffer-write-attempts>100</buffer-write-attempts>
            <buffer-write-timeout>100</buffer-write-timeout>
        </write-behind>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.13 buffer-write-timeout

Use the `buffer-write-timeout` element to define the time in milliseconds that the user thread will wait before aborting an attempt to write to the store buffer. The attempt to write to the store buffer fails only in case the buffer is full. After the time out, further attempts may be made to write to the buffer based on the value of buffer-write-attempts. This element may be changed dynamically.

This element has no child elements and no attributes.

The following example shows how to use the `buffer-write-timeout` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-behind>
            <work-manager-name>JettyWorkManager</work-manager-name>
            <batch-size>100</batch-size>
            <buffer-size>100</buffer-size>
            <buffer-write-attempts>100</buffer-write-attempts>
            <buffer-write-timeout>100</buffer-write-timeout>
        </write-behind>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.14 cache

Use the `cache` element to define a cache for a component. A *cache* is a temporary storage area for events, created exclusively to improve the overall performance of your Oracle Event Processing application. A cache is not necessary for the application to function correctly. Oracle Event Processing applications can optionally publish or consume events to and from a cache to increase the availability of the events and increase the performance of their applications.

This element has the following child elements and no attributes.

- name
- max-size
- eviction-policy
- time-to-live
- idle-time
- One of:
  - write-none
  - write-through
  - write-behind
- work-manager-name
- listeners

The following example shows how to use the `cache` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.15 caching-system

Use the `caching-element` element to define an Oracle Event Processing local caching system component. A *caching system* refers to a configured instance of a caching implementation. A caching system defines a named set of configured caches as well as the configuration for remote communication if any of the caches are distributed across multiple machines.

This element has the following child elements and no attributes.

- name
- cache.

The following example shows how to use the `caching-system` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.16 channel

Use the *channel* element to define a channel component. An Oracle Event Processing application contains one or more channel components that represent the physical conduit through which events flow between other types of components, such as between adapters and processors, and between processors and event beans (business logic POJOs).

This element has the following child elements and no attributes.

- name
- record-parameters
- playback-parameters
- max-size
- max-threads
- selector
- heartbeat
- partition-order-capacity.

The following example shows how to use the `channel` element in the component configuration file. In the example, the channel's unique identifier is `MatchOutputChannel`.

```
<channel>    <name>monitoring-control-stream</name>    <max-size>10000</max-size>
<max-threads>1</max-threads></channel>
```

4.17 channel (http-pub-sub-adapter Child Element)

Use the `channel` element to specify the channel that the http-pub-sub-adapter publishes or subscribes to for *all* http-pub-sub-adapters, whether they are local or remote or for publishing or subscribing. This element has no child elements and no attributes.

The following example shows how to use the `channel` element in the component configuration file. In the example, the `localPublisher` pub-sub adapter publishes to a local channel with pattern `/channel2`.

```
<http-pub-sub-adapter>
    <name>localPublisher</name>
    <server-context-path>/pubsub</server-context-path>
    <channel>/channel2</channel>
</http-pub-sub-adapter>
```

4.18 channel-name

Use the `channel-name` element to specify the name of the channel onto which events are to be injected or to which traced events are to be sent. This element has no child elements and no attributes. When the value of the active element is `false`, the `channel-name` value is ignored.

The element value must be a path to the channel in the following form and begin with a slash:

- Event tracing: /*serverID*/*appID*/*stageID*/**output**
- Event injection: /*serverID*/*appID*/*stageID*/**input**

The following example shows how to configure a processor for event tracing. The trace-parameters element's active child element specifies that tracing is on, and the channel-name element specifies the HTTP pub-sub channel to which to send traced elements.

```
<processor>
    <name>FindCrossRates</name>
    <trace-parameters>
        <active>true</active>
        <channel-name>/NonClusteredServer/fx/FindCrossRates/output</channel-name>
    </trace-parameters>
    <rules>
        <!-- Query rules omitted. -->
    </rules>
</processor>
```

4.19 client-id

Use the `client-id` element to specify the ID of the durable subscription client for an EDN adapter. This element has no child elements and no attributes.

4.20 coherence-cache-config

Use this element to define the Oracle Coherence cache configuration for a coherence-caching-system. This element has no child elements and no attributes.

The following example shows how to use the `coherence-cache-config` element in the component configuration file:

```
<coherence-caching-system>
    <name>nativeCachingSystem</name>
    <coherence-cache-config>
        applications/cache_cql/coherence/coherence-cache-
    </coherence-cache-config></coherence-caching-system>
```

4.21 coherence-caching-system

Use the `coherence-caching-system` element to define an Oracle Coherence caching system component. A *caching system* refers to a configured instance of a caching implementation. A caching system defines a named set of configured caches as well as the configuration for remote communication if any of the caches are distributed across multiple machines.

This element has the following child elements and no attributes.

- name
- coherence-cache-config
- coherence-cluster-config

The following example shows how to use the `coherence-caching-system` element in the component configuration file. In the example, the channel's unique identifier is `nativeCachingSystem`.

```
<coherence-caching-system>
    <name>nativeCachingSystem</name>
    <coherence-cache-config>
        applications/cache_cql/coherence/coherence-cache-
    </coherence-cache-config>
</coherence-caching-system>
```

4.22 coherence-cluster-config

Use the `coherence-cluster-config` element to define the Oracle Coherence cluster configuration for a coherence-caching-system. This element has no child elements and no attributes.

The following example shows how to use the `coherence-cache-config` element in the component configuration file:

```
<coherence-caching-system>
    <name>nativeCachingSystem</name>
    <coherence-cluster-config>
        applications/cluster_cql/coherence/coherence-cluster-
    </coherence-cluster-config></coherence-caching-system>
```

4.23 collect-interval

Use the `collect-interval` element to define the collection interval of an average-latency or max-latency element in a diagnostic profile. This element has the following child elements and no attributes.

- amount
- unit

The following example shows how to use the `collect-interval` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
```

```
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.24 concurrent-consumers

Use the `concurrent-consumers` element to define the number of consumers to create. Default value is 1. This element has no child elements and no attributes.

If you set this value to a number greater than one, be sure that your converter bean is thread-safe because the converter bean will be shared among the consumers. If the value of `concurrent-consumers` is greater than 1 and you want all the consumers to be run concurrently, then consider how you configure the work-manager you associate with this JMS adapter.

- If the `work-manager` is shared with other components (such as other adapters and Jetty) then set the `work-manager` attribute `max-threads-constraint` greater than or equal to the `concurrent-consumers` setting.
- If the `work-manager` is not shared (that is, it is dedicated to this inbound JMS adapter only) then set the `work-manager` attribute `max-threads-constraint` equal to the `concurrent-consumers` setting.

For more information, see work-manager.

The following example shows how to use the `concurrent-consumers` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.25 configuration

Use the `configuration` element to group key and value pairs. This element has the following child elements and no attributes:

- config-name
- config-value.

4.26 config-name

Use the `config-name` element to specify the key name for a configuration setting. This element has no child elements and no attributes.

4.27 config-value

Use the `config-value` element to specify the value for a configuration setting. This element has no child elements and no attributes.

4.28 connection-jndi-name

Use the optional `connection-jndi-name` element to define a JNDI name of the JMS connection factory. Default value is `weblogic.jms.ConnectionFactory` for Oracle Event Processing server JMS.

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

The following example shows how to use the `connection-jndi-name` element in the component configuration file:

4.29 connection-encrypted-password

Use the `connection-encrypted-password` element to define the encrypted jms-adapter password that Oracle Event Processing uses when it acquires a connection to the JMS service provider. This element has no child elements and no attributes.

When Oracle Event Processing calls the `createConnection` method on the `javax.jms.ConnectionFactory` to create a connection to the JMS destination (JMS queue or topic), it uses the connection-user and connection-password or `connection-encrypted-password` element, if configured. Otherwise, Oracle Event Processing uses the user and password (or encrypted-password) elements. Use either `connection-encrypted-password` or connection-password but not both.

The following example shows how to use the `connection-encrypted-password` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
    <connection-user>wlevscon</user>
    <encrypted-password>{Salted-3DES}s4YUEvH4Wl2DAjb45iJnrw==</encrypted-password>
</http-pub-sub-adapter>
```

4.30 connection-password

Use the `connection-password` element to define the jms-adapter password that Oracle Event Processing uses when it acquires a connection to the JMS service provider. This element has no child elements and no attributes.

When Oracle Event Processing calls the `createConnection` method on the `javax.jms.ConnectionFactory` to create a connection to the JMS destination (JMS queue or topic), it uses the connection-user and `connection-password` or connection-encrypted-password element, if configured. Otherwise, Oracle Event Processing uses the user and password (or encrypted-password) elements.

Use either `connection-password` or connection-encrypted-password but not both.

The following example shows how to use the `connection-password` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
    <connection-user>wlevscon</user>
    <connection-password>wlevscon</password>
</http-pub-sub-adapter>
```

4.31 connection-user

Use the `connection-user` element to define the jms-adapter user name that Oracle Event Processing uses when it acquires a connection to the JMS service provider. This element has no child elements and no attributes.

When Oracle Event Processing calls the `createConnection` method on the `javax.jms.ConnectionFactory` to create a connection to the JMS destination (JMS queue or topic), it uses the `connection-user` and connection-password or connection-encrypted-password element, if configured. Otherwise, Oracle Event Processing uses the user and password (or encrypted-password) elements.

You can use the `connection-user` and `connection-password` (or `connection-encrypted-password`) settings in applications where one security provider is used for JNDI access and a separate security provider is used for JMS access.

The following example shows how to use the `connection-user` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
    <connection-user>wlevscon</user>
    <connection-password>wlevscon</password>
</http-pub-sub-adapter>
```

4.32 context-path

Use the `context-path` element to specify the location on the web server where the application will be deployed. The `context-path` element has no attributes and no children.

The following example shows how to use the `context-path` element in a configuration file.

```
<rest-adapter>
   <name>httpInbound</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <context-path>/testhttpadapter</context-path>
</rest-adapter>
<jaxb-mapper>
   <name>xmlMapperBean</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <metadata>external_metadata_case1.xml</metadata>
</jaxb-mapper>
<csv-mapper>
   <name>csvMapperBean</name>
   <context-path>org.callcenter</context-path>
  <validate>false</validate>
</csv-mapper>
```

4.33 csv-adapter

Use a CSV Inbound adapter to accept data in the form of comma-separated values entering the EDN, and use a CSV Outbound adapter to send data in comma-separated values out of the EDN. The `csv-adapter` element has the following child elements and no attributes.

- event-interval
- initial-delay
- append
- source-url
- output-file

The following configuration file example shows how to use the `csv-adapter` element to create CSV inbound and outbound adapters.

```
<csv-adapter>
    <name>StockTradeCSVInboundAdapter</name>
</csv-adapter>

<csv-adapter>
    <name>StockTradeCSVOutboundAdapter</name>
</csv-adapter>
```

4.34 csv-mapper

Use a `csv-mapper` element to handle incoming CSV data This element has the following child elements and no attributes.

- event-type-name

- context-path
- metadata
- validate
- schema
- event-interval
- initial-delay

The following example shows how to use the `csv-mapper` element in a configuration file.

```
<rest-adapter>
   <name>httpInbound</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <context-path>/testhttpadapter</context-path>
</rest-adapter>
<jaxb-mapper>
   <name>xmlMapperBean</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <metadata>external_metadata_case1.xml</metadata>
</jaxb-mapper>
<csv-mapper>
   <name>csvMapperBean</name>
   <context-path>org.callcenter</context-path>
  <validate>false</validate>
</csv-mapper>
```

4.35 database

Use the `database` element to define a database reference for an Oracle CQL processor. This element has no child elements and the following attributes.

Table 4-1 Attributes of the database Component Configuration Element

Attribute	Description	Data Type	Required?
`name`	Unique identifier for this query.	`String`	Yes
`data-source-name`	The name of the data source as defined in the Oracle Event Processing server file.	`String`	Yes

The following example shows how to use the `database` element in the component configuration file:

```
<processor>
    <name>proc</name>
    <rules>
        <rule id="rule1"><![CDATA[
            SELECT symbol, price
            FROM ExchangeEvent retain 1 event,
            StockDb ('SELECT symbol FROM Stock WHERE symbol = ${symbol}')
        ></rule>
    </rules>

    <database name="StockDb" data-source-name="StockDs" />
```

```
</processor>
```

4.36 decision-function

Use the decision-function element to specify the name of the OBR decision function you want to use. This element has no child elements and no attributes.

The following example shows how to use the `decision-function` element.

```
 <processor>
    <name>helloworldProcessor</name>
  <rules>
    <query id="helloworldRule">
      <!-- <![CDATA[ select * from helloworldInputChannel[range 10 slide 5] >
      -->
      select * from helloworldInputChannel[now]
    </query>
  </rules>
  </processor>
  <obr-adapter>
    <name>OBRAdapter</name>
    <dictionary-url>file:helloworld.rules</dictionary-url>
    <decision-function>handler1</decision-function>
  </obr-adapter>
```

4.37 default-session

Use the `default-session` element when one or more session elements are required. All of the `session` tags by default inherit all of the elements declared in the `default-session` tag.

4.38 delivery-mode

Use the `delivery-mode` element to define the delivery mode for a jms-adapter. This element has no child elements and no attributes.

Valid delivery modes are the following:

- `persistent` (default)
- `nonpersistent`

The following example shows how to use the `delivery-mode` element in the component configuration file:

```
<jms-adapter>
    <name>jmsOutbound-map</name>
    <event-type>JMSTestEvent</event-type>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <destination-jndi-name>Topic1</destination-jndi-name>
    <delivery-mode>nonpersistent</delivery-mode>
</jms-adapter>
```

4.39 destination-jndi-name

Use the `destination-jndi-name` required element to define the JMS destination name for a jms-adapter. This element has no child elements and no attributes.

Specify either the JNDI name or the actual destination-name, but not both.

The following example shows how to use the `destination-jndi-name` element in the component configuration file:

```
<jms-adapter>
    <name>jmsOutbound-map</name>
    <event-type>JMSTestEvent</event-type>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <destination-jndi-name>Topic1</destination-jndi-name>
    <delivery-mode>nonpersistent</delivery-mode>
</jms-adapter>
```

4.40 destination-name

Use the `destination-name` required element to define the JMS destination name for a jms-adapter. This element has no child elements and no attributes.

Specify either the actual destination name or the destination-jndi-name, but not both.

The following example shows how to use the `destination-name` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.41 destination-type

Use the `destination-type` element to define the JMS destination type for a jms-adapter. This element has no child elements and no attributes.

Valid values are `TOPIC` or `QUEUE`. This property must be set to `TOPIC` whenever the durable-subscription property is set to `true`.

The following example shows how to use the `destination-type` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <destination-type>TOPIC</destination-name>
    <durable-subscription>true</durable-subscription>
    <durable-subscription-name>JmsDurableSubscription</durable-subscription-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.42 diagnostic-profiles

Use the `diagnostic-profiles` element to define one or more Oracle Event Processing diagnostic profiles. This element has the following child elements and no attributes.

- name

- profile

The following example shows how to use the `diagnostics-profiles` element in the component configuration file. In the example, the channel's unique identifier is `myDiagnosticProfiles`.

```
<diagnostics-profiles>
    <name>myDiagnosticProfiles</name>
    <profile>
        ...
    </profile>
</diagnostics-profiles>
```

4.43 dictionary-url

Use the dictionary-url element to specify the path to the OBR dictionary file that contains the rules and decision function you want to use. This element has no child elements and no attributes.

The following example shows how to use the `dictionary-url` element.

```
 <processor>
    <name>helloworldProcessor</name>
  <rules>
    <query id="helloworldRule">
      <!-- <![CDATA[ select * from helloworldInputChannel[range 10 slide 5] >
      -->
      select * from helloworldInputChannel[now]
    </query>
  </rules>
  </processor>
  <obr-adapter>
    <name>OBRAdapter</name>
    <dictionary-url>file:helloworld.rules</dictionary-url>
    <decision-function>handler1</decision-function>
  </obr-adapter>
```

4.44 direction

Use the `direction` element to define the direction for a diagnostic profile end-location or start-location. This element has no child elements and no attributes.

Valid values are:

- `INBOUND`
- `OUTBOUND`

The following example shows how to use the `direction` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
```

```
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.45 durable-subscription

Use the `durable-subscription` element to specify whether the JMS topic subscription of a jms-adapter is durable, meaning that it can persist even if subscribers become inactive. Valid values are `true` or `false`. This property is only valid if destination-type is set to `TOPIC`.

This element has no child elements and no attributes.

The following example shows how to use the `durable-subscription` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <destination-type>TOPIC</destination-name>
    <durable-subscription>true</durable-subscription>
    <durable-subscription-name>JmsDurableSubscription</durable-subscription-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.46 durable-subscription-name

Use the `durable-subscription-name` element to specify the name to uniquely identify a durable subscription of a jms-adapter. A durable subscription can persist even if subscribers become inactive. This element has no child elements and no attributes.

The following example shows how to use the `durable-subscription-name` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <destination-type>TOPIC</destination-name>
    <durable-subscription>true</durable-subscription>
    <durable-subscription-name>JmsDurableSubscription</durable-subscription-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.47 duration

Use this element to define a time duration for a schedule-time-range-offset or time-range-offset element in the following form where: `HH` is a number of hours, `MM` is a number of minutes, and `SS` is a number of seconds.

```
HH:MM:SS
```

This element has no child elements and no attributes.

The following example shows how to use the `duration` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range-offset>
        <start>2010-01-20T05:00:00</start>
        <duration>03:00:00</duration>
    </time-range-offset>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.48 edl-file

The name of the requited EDL file that is loaded into an EDN adapter. The EDL file with the specified name must be packaged in an application bundle in the `META-INF/wlevs/edn/` directory. This element has no child elements and no attributes.

The following example shows how to use the `edl-file` element.

```
<edn-adapter>
  <name>ednOutboundAdapterJaxb</name>
  <edl-file>CallCenterEvents.edl</edl-file>
  <schema-file>callcenter.xsd</schema-file>
  <validate>true</validate>
  <package-name>org.callcenter</package-name>
  <raw-xml-content>false</raw-xml-content>
  <jndi-provider-url>vm://localhost</jndi-provider-url>
  <jndi-factory>org.apache.activemq.jndi.ActiveMQInitialContextFactory
    </jndi-factory>
  <user>test1</user>
  <password>test123</password>
  <delivery-mode>nonpersistent</delivery-mode>
</edn-adapter>
```

4.49 edn-adapter

Use the `edn-adapter` element to interface with an Oracle SOA Suite event network. Use an EDN inbound adapter to receive incoming data from the Oracle SOA Suite

event network, and use an EDN outbound adapter to send outbound data to the Oracle SOA Suite event network.

This element has the following child elements and no attributes. The `client-id` and `durable-subscription-name` elements apply to the EDN input adapter only. The rest apply to both the input and output EDN adapters.

All the elements in the list except `client-id` and `durable-subscription-name` apply to both the input and output EDN adapters. The `client-id` and `durable-subscription-name` elements are unique to the input EDN adapter.

- client-id
- edl-file
- package-name
- priority
- raw-xml-content
- schema-file
- validate
- metadata
- jndi-provider-url.
- jndi-factory
- user

  One of:

  – password

  – encrypted-password
- delivery-mode
- time-to-live
- durable-subscription-name

The following example shows an EDN inbound adapter configuration. The inbound EDN adapter loads an EDL file that contains the definition of the `CallCenterActivityEvent` specified in the assembly file. The schema file is provided for schema validation. The package name is used for schema-generated JAXB classes. Raw XML content means that the EDN XML transmission is represented as ras XML. The JNDI provider and factory enable Oracle SOA Suite to locate the EDN.

```
<edn-adapter>
  <name>ednInboundAdapterJaxb</name>
  <edl-file>CallCenterEvents.edl</edl-file>
  <schema-file>callcenter.xsd</schema-file>
  <validate>true</validate>
  <package-name>org.callcenter</package-name>
  <raw-xml-content>false</raw-xml-content>
  <jndi-provider-url>vm://localhost</jndi-provider-url>
  <jndi-factory>org.apache.activemq.jndi.ActiveMQInitialContextFactory
```

```
    </jndi-factory>
  <user>test1</user>
  <password>test123</password>
</edn-adapter>
```

4.50 enabled

Use this element to define whether or not a diagnostic profile is enabled. This element has no elements and no attributes.

Valid values are:

- `true`
- `false`

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.51 encrypted-password

use the `encrypted-password` element to create a password that is changed to another form to prevent hacking. This element has no child elements and no attributes.

Use the `encrypted-password` element in the following parent elements:

- http-pub-sub-adapter: Use the `encrypted-password` element to define the encrypted password if the HTTP pub-sub server to which the Oracle Event Processing application is publishing requires user authentication.
- jms-adapter: When Oracle Event Processing acquires the JNDI `InitialContext`, it uses the user and password (or `encrypted-password`) elements. When Oracle Event Processing calls the `createConnection` method on the `javax.jms.ConnectionFactory` to create a connection to the JMS destination (JMS queue or topic), it uses the connection-user and connection-password (or connection-encrypted-password element), if configured. Otherwise, Oracle Event Processing the user and password (or `encrypted-password`) elements.
- rmi-adapter: Use the `encrypted-password` element to define an encrypted user password for RMI authentication to the Oracle WebLogic Server.

- edn-adapter: Use the `encrypted-password` element to define an encrypted user password for Oracle SOA Suite EDN or JMS authentication.

  Note:

  Use either `encrypted-password` or password but not both.

The following example shows how to use the `encrypted-password` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <encrypted-password>{Salted-3DES}s4YUEvH4Wl2DAjb45iJnrw==</encrypted-password>
</http-pub-sub-adapter>
```

4.52 end

Use the `end` element to define an end time for a time-range or schedule-time-range element. This element has no child elements and no attributes.

Express the end time as an XML Schema `dateTime` value of the form:

```
yyyy-mm-ddThh:mm:ss
```

For example, to specify that play back should start on January 20, 2010, at 5:00am and end on January 20, 2010, at 6:00 pm, enter the following:

```
  <time-range>
    <start>2010-01-20T05:00:00</start>
    <end>2010-01-20T18:00:00</end>
  </time-range>
```

For complete details of the XML Schema `dateTime` format, see `http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation`.

The following example shows how to use the `end` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range>
        <start>2010-01-20T05:00:00</start>
        <end>2010-01-20T18:00:00</end>
    </time-range>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.53 end-location

Use the `end-location` element to define the end location of a average-latency or max-latency element in a diagnostic profile. This element has the following child elements and no attributes.

- application
- stage
- direction

The following example shows how to use the `end-location` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.54 event-bean

Use the `event-bean` element to define an event bean component. This element has the following child elements and no attributes.

- name
- record-parameters
- playback-parameters

The following example shows how to use the `event-bean` element in the component configuration file. In the example, the channel's unique identifier is `myEventBean`.

```
<event-bean>
    <name>myEventBean</name>
</event-bean>
```

4.55 event-interval

Use the `event-interval` element on a CSV inbound adapter to specify the time in selected units between reading events. Possible units are nanoseconds,

microseconds, milliseconds, and seconds. This element has no child elements and no attributes.

The interval to wait between the events. Possible units are nanoseconds, microseconds, milliseconds, and seconds. For example, if the event interval is set to 1 second, then the CSV input adapter waits 1 second before processing every event. Note that the actual wait time will be slightly higher than the event interval because there is some associated overhead for thread handling. Therefore avoid very low event intervals.

The following code shows a CSV adapter with an event interval of 25 nanoseconds and an initial delay of 40 seconds. Possible units are nanoseconds, microseconds, milliseconds, and seconds.

```
<csv-adapter>
  <name>csv-inbound-adapter</name>
  <event-interval units="nanoseconds">25</event-interval>
  <initial-delay units="seconds">40</initial-delay>
</csv-adapter>
```

4.56 event-type

Use the event-type element to define the data contained in an event. This element has no child elements and no attributes.

Use the event-type element in the following parent elements:

- http-pub-sub-adapter:
  - Publishing: Optional. For both local and remote HTTP pub-sub adapters for publishing, specify the fully qualified class name of the JavaBean event to limit the types of events that are published. Otherwise, all events sent to the HTTP pub-sub adapter are published.
  - Subscribing: Required. For both local and remote HTTP pub-sub adapters for subscribing, specify the fully qualified class name of the JavaBean to which incoming messages are mapped. At runtime, Oracle Event Processing uses the incoming key-value pairs in the message to map the message data to the specified event type.

  You must register this class in the EPN assembly file as a wlevs:event-type-repository element wlevs:class child element.
- jms-adapter: Use the event-type element to specify an event type whose properties match the JMS message properties. Specify this child element only if you want Oracle Event Processing to automatically perform the conversion between JMS messages and events. If you have created your own custom converter bean, then do not specify this element.
- record-parameters: Use the event-type element to specify an event that you want to record.

The value of the event-type element must be one of the event types you defined in your event type repository.

The following example shows how to use the event-type element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
```

```
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.57 event-type-name

Use the `event-type-name` element to specify the name of an event type. An event type is a data structure that defines the data contained in an event.

The `event-type-name` element has no children and no attributes. The following example shows how to use this element in a configuration file.

```
<rest-adapter>
   <name>httpInbound</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <context-path>/testhttpadapter</context-path>
</rest-adapter>
<jaxb-mapper>
   <name>xmlMapperBean</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <metadata>external_metadata_case1.xml</metadata>
</jaxb-mapper>
<csv-mapper>
   <name>csvMapperBean</name>
   <context-path>org.callcenter</context-path>
  <validate>false</validate>
</csv-mapper>
```

4.58 eviction-policy

Use the `eviction-policy` element to define the eviction policy the cache uses when max-size is reached. This element has no child elements and no attributes.

Valid values are:

- `FIFO`: first in, first out.
- `LRU`: least recently used
- `LFU`: least frequently used (default)
- `NRU`: not recently used

The following example shows how to use the `eviction-policy` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.59 fail-when-rejected

Use the `fail-when-rejected` element to specify whether an exception of type `com.bea.wlevs.ede.api.EventProcessingException` should be raised if the event queue is full when the offer time out expires. If set to `false` or not set at all, then the event is dropped rather than an exception raised. This configuration is only applicable for event queues whose max-size value is greater than 0. The default value is `false`. For more on setting the offer time out, see offer-timeout.

This element has no child elements and no attributes.

In the following example, the channel is configured to raise an `EventProcessingException` if 15 seconds pass while the event queue is full.

```
<channel>
    <name>QueuedChannel</name>
    <max-size>1000</max-size>
    <max-threads>1</max-threads>
    <offer-timeout>15000000000</offer-timeout>
    <fail-when-rejected>true</fail-when-rejected>
</channel>
```

4.60 group-binding

Edit the component configuration file to add `group-binding` child elements to the `jms-adapter` element for the JMS inbound adapters. Add one `group-binding` element for each possible JMS message-selector value. For more information, see bindings (jms-adapter).

This element has the param child element and the group-id attribute. The group-id attribute is required, is type `String`, and is the name of a cluster element that groups child elements.

The following example shows how to use the `group-binding` element in the component configuration file:

```
<jms-adapter>
    <name>JMSInboundAdapter</name>
    <event-type>StockTick</event-type>
    <jndi-provider-url>t3://ppurich-pc:7001</jndi-provider-url>
    <destination-jndi-name>./Topic1</destination-jndi-name>
    <user>weblogic</user>
    <password>weblogic1</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>true</session-transacted>
    <message-selector>${CONDITION}</message-selector>
    <bindings>
        <group-binding group-id="ActiveActiveGroupBean_group1">
            <param id="CONDITION">acctid > 400</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group2">
            <param id="CONDITION">acctid BETWEEN 301 AND 400</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group3">
            <param id="CONDITION">acctid BETWEEN 201 AND 300</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group4">
            <param id="CONDITION">acctid <= 200</param>
        </group-binding>
     </bindings>
</jms-adapter>
```

4.61 heartbeat

Use the `heartbeat` element to define a new heartbeat time out for a system-time stamped channel component. By default, the time out value is 100 milliseconds, which is 100,000,000 nanoseconds. This element has no child elements and no attributes. A heartbeat checks for the arrival of data on the channel every so many seconds

Oracle Event Processing generates a heartbeat on a system time stamped relation or stream channel when no data has arrived on the channel for more than the value for this setting.

The heartbeat child element applies to system-time stamped relations or streams only when no events arrive in the event channels that are feeding the processors and the processor has been configured with a statement that includes some temporal operator, such as a time-based window or a pattern matching with duration.

The following example shows how to use the `heartbeat` element in the component configuration file. In the example, the channel's unique identifier is `MatchOutputChannel`.

```
<channel>
    <name>MatchOutputChannel</name>
    <max-size>0</max-size>
    <max-threads>0</max-threads>
    <selector>match</selector>
    <heartbeat>10000</heartbeat>
</channel>
```

4.62 http-pub-sub-adapter

Use the http-pub-sub-adapter element to define an HTTP publish-subscribe server adapter component.

This element has the following child elements and no attributes.

- name
- record-parameters
- playback-parameters
- work-manager-name
- netio
- One of:
  - server-context-path
  - server-url
- event-type
- user
- One of:
  - password
  - encrypted-password

The following example shows how to use the `http-pub-sub-adapter` element in the component configuration file. In the example, the adapter's unique identifier is `remotePub`.

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
</http-pub-sub-adapter>
```

4.63 idle-time

Use the `idle-time` element to define the number of milliseconds a cache entry may not be accessed before being actively removed from the cache. By default, there is no idle-time set. This element may be changed dynamically. This element has no child elements and no attributes.

The following example shows how to use the `idle-time` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.64 initial-delay

Use the `initial-delay` element on an adapter to specify the time in nanoseconds before beginning to read events. The `initial-delay` element has no children and no attributes.

The following code shows a CSV adapter with an event interval of 25 `nanoseconds` and an initial delay of 40 seconds. Possible units are nanoseconds, `microseconds`, `milliseconds`, and `seconds`.

```
  <csv-adapter>
    <name>csv-inbound-adapter</name>
    <event-interval units="nanoseconds">25</event-interval>
    <initial-delay units="seconds">40</initial-delay>
  </csv-adapter>
```

4.65 inject-parameters

Use the `inject-parameters` element to configure event injection for a stage in the event processing network. This element has the following child elements and no attributes:

- channel-name
- active

The component configuration excerpt shown in the following example illustrates how you might configure a processor for event injection. The inject-parameters element's active child element specifies that injection is on, while the channel-name element specifies the HTTP pub-sub channel to which injected elements should be sent.

```
<processor>
    <name>FindCrossRates</name>
    <inject-parameters>
        <active>true</active>
        <channel-name>/NonClusteredServer/fx/FindCrossRates/input</channel-name>
    </inject-parameters>
    <rules>
        <!-- Query rules omitted. -->
    </rules>
</processor>
```

4.66 jaxb-mapper

The `jaxb-mapper` element to handle incoming XML and JSON data This element has the following child elements and no attributes.

Child Elements

- event-type-name
- context-path
- metadata
- validate
- schema

Example

The following example shows how to use the `jaxb-mapper` element in a configuration file.

```
<rest-adapter>
   <name>httpInbound</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <context-path>/testhttpadapter</context-path>
</rest-adapter>
<jaxb-mapper>
   <name>xmlMapperBean</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <metadata>external_metadata_case1.xml</metadata>
</jaxb-mapper>
<csv-mapper>
   <name>csvMapperBean</name>
   <context-path>org.callcenter</context-path>
  <validate>false</validate>
</csv-mapper>
```

4.67 jms-adapter

Use the `jms-adapter` element to define a JMS adapter component. This element has the following child elements and no attributes.

Chile Elements

- name
- record-parameters
- playback-parameters
- event-type
- jndi-provider-url
- connection-jndi-name
- One of:
  - destination-jndi-name
  - destination-name
- user

  One of:
  - password
  - encrypted-password
- connection-user

  One of:
  - connection-password
  - connection-encrypted-password
- work-manager
- concurrent-consumers
- message-selector
- session-ack-mode-name
- session-transacted
- delivery-mode

Example

The following example shows how to use the `jms-adapter` element in the component configuration file. In the example, the adapter's unique identifier is `jmsInbound-text`.

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.68 jndi-factory

Use the optional `jndi-factory` element to define a JNDI factory for a jms-adapter or rmi-adapter. The default JNDI factory name value for Oracle Event Processing server JMS is `weblogic.jndi.WLInitialContextFactory`.

This element has no child elements and no attributes.

The following example shows how to use the `jndi-factory` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <jndi-factory>weblogic.jndi.WLInitialContextFactory</jndi-factory>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.69 jndi-name

Use the `jndi-name` element to specify the Java Naming and Directory Interface (JNDI) name of the component your application needs to locate and access over a Remote Method Invocation (RMI) input or output adapter.

The following example shows how to use the `jndi-name` element.

```
<rmi-adapter>
    <name>rmi-outbound-adapter</name>
    <jndi-name>RMIOutboundJNDIName</jndi-name>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <jndi-factory>weblogic.jndi.WLInitialContextFactory</jndi-factory>
 </rmi-adapter>
```

4.70 jndi-provider-url

Use the `jndi-provider-url` required element to define a JNDI provider URL for a jms-adapter. This element has no child elements and no attributes.

The following example shows how to use the `jndi-provider-url` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
```

```
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.71 listeners

Use the `listeners` element to define the behavior for cache listeners. This element has the work-manager-name child element and the asynchronous attribute.

The work-manager-name child element specifies the work manager to use to invoke listeners asynchronously. Oracle Event Processing ignores the work manager value when you enable synchronous invocations. If a work manager is specified for the cache itself, this value overrides it for invoking listeners only.

The asynchronous attribute is required, is type `Boolean`, and when `true`, executes listeners asynchronously. The default value is `false`.

The following example shows how to use the `listeners` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-behind>
            <work-manager-name>JettyWorkManager</work-manager-name>
            <batch-size>100</batch-size>
            <buffer-size>100</buffer-size>
            <buffer-write-attempts>100</buffer-write-attempts>
            <buffer-write-timeout>100</buffer-write-timeout>
        </write-behind>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.72 location

Use the `location` element to define the location of a throughput element in a diagnostic profile. This element has the following child elements and no attributes.

- application
- stage

  direction.

The following example shows how to use the `location` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <throughput>
            <throughput-interval>
                <amount>100000</amount>
```

```
                <unit>MICROSECONDS</unit>
            </throughput-interval>
            <average-interval>
                <amount>100000000</amount>
                <unit>NANOSECONDS</unit>
            </average-interval>
            <location>
                <application>diagnostic</application>
                <stage>AlertEventStream</stage>
                <direction>INBOUND</direction>
            </location>
        </throughput>
    </profile>
</diagnostic-profiles>
```

4.73 max-latency

Use the `max-latency` element to define the maximum latency calculation of a diagnostic profile. This element has the following child elements and no attributes.

- name
- collect-interval
- start-location
- end-location

The following example shows how to use the `max-latency` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.74 max-size

Use the `max-size` element in the following parent elements:

- channel: Use the `max-size` child element to specify the maximum size of the channel. Zero-size channels synchronously pass-through events. Non-zero size channels process events asynchronously and buffer events by the requested size. If max-threads is zero, then `max-size` is zero. The default value is 0.
- cache: Use the `max-size` element to define the number of cache elements in memory after which eviction or paging occurs. Currently, the maximum cache size is 2^31-1 entries. This element may be changed dynamically

This element has no child elements and no attributes.

The following example shows how to use the `max-size` element in the component configuration file:

```
<stream>
    <name>monitoring-control-stream</name>
    <max-size>10000</max-size>
    <max-threads>1</max-threads>
</stream>
```

4.75 max-threads

Use the `max-threads` element to define the maximum number of threads that Oracle Event Processing server uses to process events for a channel. This element has no child elements and no attributes.

The default value is 0. If the `max-size` attribute is 0, then setting a value for `max-threads` has no effect.

When set to 0, the channel acts as a pass-through. When `max-threads > 0`, the channel acts as classic blocking queue, where upstream components are producers of events and the downstream components are the consumers of events. The queue size is defined by the configuration max-size. There will be up to `max-threads` number of threads consuming events from the queue.

You can change `max-threads` from 0 to a positive integer (that is, from a pass through to multiple threads) without redeploying. However, if you change `max-threads` from a positive integer to 0 (that is, from multiple threads to a pass through), then you must redeploy your application. Setting this value has no effect when max-size is 0.

The following example shows how to use the `max-threads` element in the component configuration file:

```
<channel>
    <name>monitoring-control-stream</name>
    <max-size>10000</max-size>
    <max-threads>1</max-threads>
</channel>
```

4.76 message-selector

Use the `message-selector` element with a JMS message selector to filter messages in a jms-adapter. This element has no child elements and no attributes.

The syntax of a message selector expression is based on a subset of the SQL92 conditional expression syntax and message headers and properties. For example, to select messages based on property `EventType`, you could use:

```
EventType = 'News' OR 'Commentary'
```

The following example shows how to use the `message-selector` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <message-selector>EventType = 'News' OR 'Commentary'</message-selector>
```

```
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.77 metadata

Use the `metadata` element to specify the name of the file that contains the JAXB Boxy external metadata for customizing the JAXB mapping. This element is optional and has no child elements and no attributes.

The following example shows how to use the metadata element in a configuration file.

```
<rest-adapter>
   <name>httpInbound</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <context-path>/testhttpadapter</context-path>
</rest-adapter>
<jaxb-mapper>
   <name>xmlMapperBean</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <metadata>external_metadata_case1.xml</metadata>
</jaxb-mapper>
<csv-mapper>
   <name>csvMapperBean</name>
   <context-path>org.callcenter</context-path>
  <validate>false</validate>
</csv-mapper>
```

4.78 name

The name of the element. Use the `name` element in the following parent elements:

- EPN components: Use the `name` element to associate this application configuration element with its corresponding element in the assembly file. Valid value is the corresponding assembly file `id` attribute.
- diagnostic-profiles: Use the `name` element to uniquely identify the `diagnostic-profiles` element and each of its `profile` child elements.
- parameter: Use the `name` element to define the name of a name and value pair.

This element has no child elements and no attributes.

The following example shows how to use the `name` element in the component configuration file. In the example, the channel's unique identifier is `myDiagnosticProfiles`.

```
<diagnostics-profiles>
    <name>myDiagnosticProfiles</name>
    <profile>
        ...
    </profile>
</diagnostics-profiles>
```

4.79 netio

Use the `netio` element to define a network input/output port for a component.

Note:

When a child of the `adapter` element, this element is for internal use only.

This element has the following child elements and no attributes.

- provider-name
- num-threads
- accept-backlog

The following example shows how to use the `netio` element in the component configuration file:

```
<netio>
        <provider-name>providerCache</provider-name>
        <num-threads>1000</num-threads>
</netio>
```

4.80 num-threads

Use the `num-threads` element to define the number of threads in a network input/output port for a component. This element has no child elements and no attributes.

The following example shows how to use the `num-threads` element in the component configuration file:

```
<netio>
        <provider-name>providerCache</provider-name>
        <num-threads>1000</num-threads>
</netio>
```

4.81 obr-adapter

Use an `obr-adapter` adapter to treat events as Java code so that you can evaluate Oracle Business Rules (OBR) and trigger events based on the findings. With an OBR adapter, you can add OBR logic to a CQL processor downstream.

This element has the following child elements and no attributes.

- decision-function
- dictionary-url

The following example shows an OBR adapter configuration.

```
 <processor>
    <name>helloworldProcessor</name>
  <rules>
    <query id="helloworldRule">
      <!-- <![CDATA[ select * from helloworldInputChannel[range 10 slide 5] >
      -->
      select * from helloworldInputChannel[now]
    </query>
  </rules>
  </processor>
  <obr-adapter>
    <name>OBRAdapter</name>
    <dictionary-url>file:helloworld.rules</dictionary-url>
    <decision-function>handler1</decision-function>
  </obr-adapter>
```

4.82 offer-timeout

Use the `offer-timeout` element to specify the amount of time, when an event queue is full and a pending insert (the offer) is blocked, after which the pending event will be dropped or an exception raised. An exception will be raised when the fail-when-rejected value is set to true; otherwise, the event will be dropped. This setting is only applicable for event queues whose max-size value is greater than 0. The `offer-timeout` value should be specified as nanoseconds. The default is 60 seconds.

This element has no child elements and no attributes.

In the following example, the channel is configured to raise an `EventProcessingException` if 15 seconds pass while the event queue is full.

```
<channel>
    <name>QueuedChannel</name>
    <max-size>1000</max-size>
    <max-threads>1</max-threads>
    <offer-timeout>15000000000</offer-timeout>
    <fail-when-rejected>true</fail-when-rejected>
</channel>
```

4.83 output-file

Use the `output-file` entry to specify where a CSV Outbound adapter saves the outbound event data. This element has no child elements or attributes.

You can use the append element to specify whether to create a new file, overwrite an existing file, or append to an existing file.

The following example shows the assembly file entry for StockTradeCSVOutboundAdapter.

```
<wlevs:adapter id="StockTradeCSVOutboundAdapter" provider="csv-outbound">
    <wlevs:instance-property name="eventType" value="TradeEvent"/>
    <wlevs:instance-property name="outputFile"
        value="/scratch/mpawlan/oep9-19/oep/utils/load-generator/StockData.csv"/>
    <wlevs:instance-property name="append" value="false"/>
</wlevs:adapter>
```

4.84 package-name

The optional package name of the schema-generated classes for JAXB. When set, the specified package name overrides the package name inferred from the EDL file. This element has no child elements and no attributes.

The following example shows how to use the `package-name` element.

```
<edn-adapter>
  <name>ednOutboundAdapterJaxb</name>
  <edl-file>CallCenterEvents.edl</edl-file>
  <schema-file>callcenter.xsd</schema-file>
  <validate>true</validate>
  <package-name>org.callcenter</package-name>
  <raw-xml-content>false</raw-xml-content>
  <jndi-provider-url>vm://localhost</jndi-provider-url>
  <jndi-factory>org.apache.activemq.jndi.ActiveMQInitialContextFactory
    </jndi-factory>
  <user>test1</user>
  <password>test123</password>
```

```
  <delivery-mode>nonpersistent</delivery-mode>
</edn-adapter>
```

4.85 param

Use the `param` element to associate a message selector value with the parameter name specified in the `message-selector` element. This element has no child elements and the `id` attribute. The id attribute is required, is type `String`, and identifies the value to pass to the `message-selector` element.

For more information, see bindings (jms-adapter).

The following example shows how to use the `param` element in the component configuration file. When an application with this configuration is deployed to an Oracle Event Processing server with a `cluster` element `groups` child element that contains `ActiveActiveGroupBean_group1`, the `CONDITION` parameter is `acctid > 400`. The application processes events with an `acctid` property that is greater than 400.

```
<jms-adapter>
    <name>JMSInboundAdapter</name>
    <event-type>StockTick</event-type>
    <jndi-provider-url>t3://ppurich-pc:7001</jndi-provider-url>
    <destination-jndi-name>./Topic1</destination-jndi-name>
    <user>weblogic</user>
    <password>weblogic1</password>
    <work-manager>JettyWorkManager</work-manager>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>true</session-transacted>
    <message-selector>${CONDITION}</message-selector>
    <bindings>
        <group-binding group-id="ActiveActiveGroupBean_group1">
            <param id="CONDITION">acctid > 400</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group2">
            <param id="CONDITION">acctid BETWEEN 301 AND 400</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group3">
            <param id="CONDITION">acctid BETWEEN 201 AND 300</param>
        </group-binding>
        <group-binding group-id="ActiveActiveGroupBean_group4">
            <param id="CONDITION">acctid <= 200</param>
        </group-binding>
     </bindings>
</jms-adapter>
```

4.86 parameter

Use the `parameter` element to define a `name` and `value` parameter for a component. This element has the following child elements and no attributes.

- name
- value

The following example shows how to use the `parameter` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
```

```
        <store-policy-parameters>
            <parameter>
                <name>timeout</name>
                <value>300</value>
            <parameter>
        </store-policy-parameters>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.87 params

Use the `params` element to define the parameters for a bindings (processor) element. The value of this element is a comma-separated list of simple type values. The order of the values must correspond with the order of the parameters in the Oracle CQL rule associated with this `binding`.

This element has no child elements and the `id` attribute. The `id` attribute is not required, is type `String`, and is the unique identifier for the `params` element.

The following example shows how to use the `params` element in the component configuration file:

```
<processor>
    <name>processor1</name>
        <record-parameters>
            <dataset-name>test1data</dataset-name>
            <event-type-list>
                <event-type>SimpleEvent</event-type>
            </event-type-list>
            <provider-name>test-rdbms-provider</provider-name>
            <batch-size>1</batch-size>
            <batch-time-out>10</batch-time-out>
        </record-parameters>
        <rules>
            <rule id="rule1"><![CDATA[
                select stockSymbol, avg(price) as percentage
                from StockTick retain 5 events
                where stockSymbol=?
                having avg(price)  > ? or avg(price)  < ?
            ></rule>
        </rules>
        <bindings>
            <binding id="rule1">
                <params>BEAS,10.0,-10.0</params>
                <params id="IBM">IBM,5.0,5.0</params>
            </binding>
        </bindings>
</processor>
```

4.88 partition-order-capacity

Use the `partition-order-capacity` element to define the maximum capacity of a query partition when the ordering-constraint attribute is set to PARTITION_ORDERED. This element has no child elements and no attributes.

Set the `partition-order-capacity` element on a channel component when you have configured a CQL processor for parallel execution and the `ordering-constraint` attribute is set to PARTITION_ORDERED.

To have the capacity value apply at a larger scope, you can set it in the server configuration file. See Component Configuration Schema.

The following example shows how to use the `partition-order-capacity` element in the component configuration file:

```
<channel>
    <name>MatchOutputChannel</name>
    <max-size>0</max-size>
    <max-threads>0</max-threads>
    <selector>match</selector>
    <partition-order-capacity>20</partition-order-capacity>
</channel>
```

4.89 password

Use the `password` element to define a password in the following parent elements. This element has no child elements and no attributes.

- http-pub-sub-adapter: Use the `password` element to define the user password if the HTTP pub-sub server to which the Oracle Event Processing application is publishing requires user authentication.
- jms-adapter: When Oracle Event Processing acquires the JNDI `InitialContext`, it uses the user and `password` (or encrypted-password) elements. When Oracle Event Processing calls the `createConnection` method on the `javax.jms.ConnectionFactory` to create a connection to the JMS destination (JMS queue or topic), it uses the connection-user and connection-password (or connection-encrypted-password element), if configured. Otherwise, Oracle Event Processing the user and `password` (or encrypted-password) elements.
- rmi-adapter: Use the `password` element to define the user password for RMI authentication to the Oracle WebLogic Server.
- edn-adapter: Use the `password` element to define the user password for Oracle SOA Suite EDN or JMS authentication.

> **Note:**
>
> Use either encrypted-password or `password` but not both.

The following example shows how to use the `password` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
</http-pub-sub-adapter>
```

4.90 playback-parameters

Use the `playback-parameters` element to define event playback parameters for a component to test a component. You play the events back at a later stage in the application such as in an event bean. In the event bean, you query the events and make fixes to your application based on your findings This element has the following child elements and no attributes.

- playback-parameters
- event-type-list

- provider-name
- store-policy-parameters
- max-size
- max-threads
- One of:
  - time-range
  - time-range-offset
- One of:
  - schedule-time-range
  - schedule-time-range-offset
- playback-speed
- repeat

The following example shows how to use the `playback-parameters` element in the component configuration file:

```
<playback-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
            <event-type>TupleEvent1</event-type>
        </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
</playback-parameters>
```

4.91 priority

Use the priority element to set the JMS priority for events published to the EDN. The JMS API defines ten levels of priority value with 0 the lowest priority and 9 the highest. Clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority. Priority is set to 4 by default.This element has not child elements and no attributes.

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <priority>2</priority>
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.92 processor (Oracle CQL)

Use the `processor` element to define an Oracle CQL processor component. This element has the following child elements and no attributes.

- name

- record-parameters
- playback-parameters
- rules
- bindings (processor)

The following example shows how to use the `processor` element in the component configuration file. In the example, the unique identifier of the processor is `cqlProcessor`.

```
<processor>
    <name>cqlProcessor</name>
    <rules>
        <view id="lastEvents" schema="cusip bid srcId bidQty ask askQty seq"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from inputChannel[partition by srcId, cusip rows 1]
        ></view>
        <view id="bidask" schema="cusip bid ask"><![CDATA[
            select cusip, max(bid), min(ask)
            from lastEvents
            group by cusip
        ></view>
            <view ...><![CDATA[
                ...
        ></view>
        ...
        <view id="MAXBIDMINASK" schema="cusip bidseq bidSrcId bid askseq askSrcId ask bidQty
askQty"><![CDATA[
            select bid.cusip, bid.seq, bid.srcId as bidSrcId, bid.bid, ask.seq,
                ask.srcId as askSrcId, ask.ask, bid.bidQty, ask.askQty
            from BIDMAX as bid, ASKMIN as ask
            where bid.cusip = ask.cusip
        ></view>
        <query id="BBAQuery"><![CDATA[
            ISTREAM(select bba.cusip, bba.bidseq, bba.bidSrcId, bba.bid, bba.askseq,
                bba.askSrcId, bba.ask, bba.bidQty, bba.askQty,
                "BBAStrategy" as intermediateStrategy, p.seq as correlationId, 1 as priority
            from MAXBIDMINASK as bba, inputChannel[rows 1] as p where bba.cusip = p.cusip)
        ></query>
        <query id="MarketRule"><![CDATA[
             SELECT symbol, AVG(price) AS average, :1 AS market
             FROM StockTick [RANGE 5 SECONDS]
             WHERE symbol = :2
        ></query>
    </rules>
    <bindings>
        <binding id="MarketRule">
            <params id="nasORCL">NASDAQ, ORCL</params>
            <params id="nyJPM">NYSE, JPM</params>
            <params id="nyWFC">NYSE, WFC</params>
        </binding>
    </bindings>
</processor>
```

4.93 profile

Use the `profile` element to define a diagnostic profile. This element has the following child elements and no attributes.

- name
- enabled
- start-stage

- max-latency
- average-latency
- throughput

The following example shows how to use the `profile` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
        <average-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
            <threshhold>
                <amount>100</amount>
                <unit>MILLISECONDS</unit>
            </threshhold>
        </average-latency>
        <throughput>
            <throughput-interval>
                <amount>100000</amount>
                <unit>MICROSECONDS</unit>
            </throughput-interval>
            <average-interval>
                <amount>100000000</amount>
                <unit>NANOSECONDS</unit>
            </average-interval>
            <location>
                <application>diagnostic</application>
                <stage>AlertEventStream</stage>
                <direction>INBOUND</direction>
            </location>
        </throughput>
    </profile>
</diagnostic-profiles>
```

4.94 query

Use the `query` element to define an Oracle CQL query for a component. This element has no child elements and the following attributes.

Table 4-2 Attributes of the query Component Configuration Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this query.	`String`	Yes.
`active`	Execute this query when the application is deployed and run. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.
`ordering-constraint`	Enable or disable parallel query execution, through which events can be processed in parallel rather than serially. The attribute supports the following three values: • `ORDERED` means that the query must handle events serially. This is the default behavior. • `UNORDERED` means that, whenever possible, the CQL processor will execute in parallel on multiple threads to process the events. • `PARTITION_ORDERED` means that when the query is partitioning events, ensure total order within a partition and (if possible) disregard order across partitions.	String	No.
`partition-expression`	The partition expression (used in the CQL code) that should be the basis for relaxing the cross-partition ordering constraint.	String	No.

The following example shows how to use the `query` element in the component configuration file:

```
<processor>
    <name>cqlProcessor</name>
    <rules>
        <view id="lastEvents" schema="cusip bid srcId bidQty ask askQty seq"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from inputChannel[partition by srcId, cusip rows 1]
        ></view>
        <view id="bidask" schema="cusip bid ask"><![CDATA[
            select cusip, max(bid), min(ask)
            from lastEvents
            group by cusip
        ></view>
            <view ...><![CDATA[
                ...
        ></view>
        ...
        <view id="MAXBIDMINASK" schema="cusip bidseq bidSrcId bid askseq askSrcId ask bidQty
askQty"><![CDATA[
            select bid.cusip, bid.seq, bid.srcId as bidSrcId, bid.bid, ask.seq,
                ask.srcId as askSrcId, ask.ask, bid.bidQty, ask.askQty
            from BIDMAX as bid, ASKMIN as ask
            where bid.cusip = ask.cusip
        ></view>
        <query id="BBAQuery"><![CDATA[
            ISTREAM(select bba.cusip, bba.bidseq, bba.bidSrcId, bba.bid, bba.askseq,
```

```
                bba.askSrcId, bba.ask, bba.bidQty, bba.askQty, "BBAStrategy" as
intermediateStrategy,
                p.seq as correlationId, 1 as priority
            from MAXBIDMINASK as bba, inputChannel[rows 1] as p where bba.cusip = p.cusip)
        ></query>
    </rules>
</processor>
```

4.95 raw-xml-content

Use the raw-xml-content element to specify whether the EDN XML data transmission should be represented as raw XML (when `true`) or as a Java object (when `false`). The default is `false`. This element has no child elements and no attributes.

The following example shows how to use the `raw-xml-content` element.

```
<edn-adapter>
  <name>ednOutboundAdapterJaxb</name>
  <edl-file>CallCenterEvents.edl</edl-file>
  <schema-file>callcenter.xsd</schema-file>
  <validate>true</validate>
  <package-name>org.callcenter</package-name>
  <raw-xml-content>false</raw-xml-content>
  <jndi-provider-url>vm://localhost</jndi-provider-url>
  <jndi-factory>org.apache.activemq.jndi.ActiveMQInitialContextFactory
    </jndi-factory>
  <user>test1</user>
  <password>test123</password>
  <delivery-mode>nonpersistent</delivery-mode>
</edn-adapter>
```

4.96 record-parameters

Use the `record-parameters` element to define event record parameters to test a component. While the application runs, you record the events that flow out of an EPN component into a persistent store. This element has the following child elements and no attributes.

- dataset-name
- event-type-list
- provider-name
- store-policy-parameters
- max-size
- max-threads
- One of:
  - time-range
  - time-range-offset
- batch-size
- batch-time-out

The following example shows how to use the `record-parameters` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.97 repeat

Use the `repeat` element to define whether or not to repeat playback-parameters. This element has no child elements and no attributes.

Valid values are:

- `true`
- `false`

The following example shows how to use the `duration` element in the component configuration file:

```
<playback-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range-offset>
        <start>2010-01-20T05:00:00</start>
        <duration>03:00:00</duration>
    </time-range-offset>
    <repeat>true</repeat>
</playback-parameters>
```

4.98 rest-adapter

Use the `rest-adapter` element to define a Representational State Transfer (REST) adapter that receives HTTP Post data from an external client through the REST protocol. A REST adapter can accept data in the following forms and convert that data into the Oracle Event Processing event configured on the inbound REST adapter: XML, CSV, and JavaScript Object Notation (JSON).

This element has the following child elements and no attributes.

- event-type-name
- context-path

The following example file shows the `rest-adapter` configuration for receiving POST data, the `jaxb-mapper` configuration for handling incoming XML and JSON data, and the `csv-mapper` configuration for handling incoming CSV data.

```
 <rest-adapter>
    <name>httpInbound</name>
    <event-type-name>CallCenterActivity</event-type-name>
```

```
   <context-path>/testhttpadapter</context-path>
 </rest-adapter>
 <jaxb-mapper>
   <name>xmlMapperBean</name>
   <event-type-name>CallCenterActivity</event-type-name>
   <metadata>external_metadata_case1.xml</metadata>
 </jaxb-mapper>
 <csv-mapper>
   <name>csvMapperBean</name>
   <context-path>org.callcenter</context-path>
  <validate>false</validate>
 </csv-mapper>
```

4.99 rest-outbound-adapter

Use the `rest-outbound-adapter` element to define a REST adapter that sends HTTP SEND data to an external client through the REST protocol.

4.100 rmi-adapter

Use the rmi-adapter element to define an RMI outbound adapter. An RMI outbound adapter writes event information to an RMI connection. This element has the following child elements and no attributes.

- jndi-name
- jndi-provider-url
- jndi-factory
- user

  One of:

  – password

  – encrypted-password

The following example shows an RMI outbound adapter configuration.

```
<rmi-adapter>
    <name>rmi-outbound-adapter</name>
    <jndi-name>RMIOutboundJNDIName</jndi-name>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <jndi-factory>weblogic.jndi.WLInitialContextFactory</jndi-factory>
 </rmi-adapter>
```

4.101 rules

Use this element to create rules for an Oracle CQL processor (Oracle CQL). This element has the following child elements and no attributes.

- rules
- query
- view

The following example shows how to use the `rules` element in the component configuration file:

```
<processor>
    <name>cqlProcessor</name>
    <rules>
        <view id="lastEvents" schema="cusip bid srcId bidQty ask askQty seq"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from inputChannel[partition by srcId, cusip rows 1]
        ></view>
        <view id="bidask" schema="cusip bid ask"><![CDATA[
            select cusip, max(bid), min(ask)
            from lastEvents
            group by cusip
        ></view>
            <view ...><![CDATA[
                ...
        ></view>
        ...
        <view id="MAXBIDMINASK" schema="cusip bidseq bidSrcId bid askseq askSrcId ask bidQty
askQty"><![CDATA[
            select bid.cusip, bid.seq, bid.srcId as bidSrcId, bid.bid, ask.seq,
                ask.srcId as askSrcId, ask.ask, bid.bidQty, ask.askQty
            from BIDMAX as bid, ASKMIN as ask
            where bid.cusip = ask.cusip
        ></view>
        <query id="BBAQuery"><![CDATA[
            ISTREAM(select bba.cusip, bba.bidseq, bba.bidSrcId, bba.bid, bba.askseq,
                bba.askSrcId, bba.ask, bba.bidQty, bba.askQty,
                "BBAStrategy" as intermediateStrategy, p.seq as correlationId, 1 as priority
            from MAXBIDMINASK as bba, inputChannel[rows 1] as p where bba.cusip = p.cusip)
        ></query>
    </rules>
</processor>
```

4.102 schema

Use the `schema` element to specify a space-delimited list of stream elements to use in the view. The `schema` element has no child elements and no attributes.

The following example shows how to use the schema element in an Oracle CQL rule.

```
<processor>
    <name>cqlProcessor</name>
    <rules>
        <view id="lastEvents" schema="cusip bid ask"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from inputChannel[partition by srcId, cusip rows 1]
        ></view>
        <view id="bidask" schema="cusip bid ask"><![CDATA[
            select cusip, max(bid), min(ask)
            from lastEvents
            group by cusip
        ></view>
            <view ...><![CDATA[
                ...
        ></view>
        ...
        <view id="MAXBIDMINASK" schema="cusip bidseq bidSrcId bid askseq askSrcId
ask bidQty askQty"><![CDATA[
            select bid.cusip, bid.seq, bid.srcId as bidSrcId, bid.bid, ask.seq,
ask.srcId as askSrcId, ask.ask, bid.bidQty, ask.askQty
            from BIDMAX as bid, ASKMIN as ask
            where bid.cusip = ask.cusip
        ></view>
        <query id="BBAQuery"><![CDATA[
            ISTREAM(select bba.cusip, bba.bidseq, bba.bidSrcId, bba.bid, bba.askseq,
bba.askSrcId, bba.ask,
```

```
                bba.bidQty, bba.askQty, "BBAStrategy" as intermediateStrategy, p.seq
as correlationId, 1 as priority
            from MAXBIDMINASK as bba, inputChannel[rows 1] as p where bba.cusip =
p.cusip)
        ></query>
    </rules>
```

4.103 schema-file

Use the `schema-file` element to specify the name of the schema file for validation. The `schema` element is required when the `validate` element is set to `true`. The `schema-file` element has no child elements and no attributes.

The following example shows how to use the `schema-file` element in a configuration file.

```
<edn-adapter>
  <name>ednInboundAdapterJaxb</name>
  <edl-file>CallCenterEvents.edl</edl-file>
  <schema-file>callcenter.xsd</schema-file>
  <validate>true</validate>
  <package-name>org.callcenter</package-name>
  <raw-xml-content>false</raw-xml-content>
  <jndi-provider-url>vm://localhost</jndi-provider-url>
  <jndi-factory>org.apache.activemq.jndi.ActiveMQInitialContextFactory
    </jndi-factory>
  <user>test1</user>
  <password>test123</password>
</edn-adapter>
```

4.104 selector

Use the `selector` element to specify which upstream Oracle CQL processor queries are permitted to send their results to a downstream channel. This element has no child elements and no attributes.

Concepts

The figure shows an EPN with `filteredStream` channel connected to the upstream `filteredFanoutProcessor` Oracle CQL processor.

The following queries are configured for the filteredFanoutProcesso Oracle CQL processor.

```
<processor>
    <name>filterFanoutProcessor</name>
    <rules>
        <query id="Yr3Sector"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from priceStream where sector="3_YEAR"
        ></query>
        <query id="Yr2Sector"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from priceStream where sector="2_YEAR"
        ></query>
        <query id="Yr1Sector"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
```

```
            from priceStream where sector="1_YEAR"
        ></query>
    </rules>
</processor>
```

When you specify more than one query for an Oracle CQL processor, then, by default, all query results are sent to the processor's out-bound channel (`filteredStream`). Optionally, in the component configuration source, you can use the `channel` element `selector` child element to specify a space-delimited list of one or more Oracle CQL query names that can send their results to the channel as shows in the following example. In the example, the query results for query `Yr3Sector` and `Yr2Sector` are sent to `filteredStream` but the query results for query `Yr1Sector` are not.

```
<channel>
    <name>filteredStream</name>
    <selector>Yr3Sector Yr2Sector</selector>
</channel>
```

You may configure a `channel` element with a `selector` before creating the queries in the upstream processor. In this case, you must specify query names that match the names in the `selector`.

Note:

The `selector` attribute is only applicable if the up-stream stage is an Oracle CQL processor.

Example

The following example shows how to use the `selector` element in the component configuration file:

```
<processor>
    <name>PatternProcessor</name>
    <rules>
        <query id="match"><![CDATA[
            select T.firstW as firstw, T.lastZ as lastz, T.price as price
            from StockInputsStream
            MATCH_RECOGNIZE (
                MEASURES A.c1 as firstW, last(Z.c1) as lastZ, A.c2 as price
                PATTERN(A W+ X+ Y+ Z+)
                DEFINE  A as A.c1 = A.c1,
                        W as W.c2 < prev(W.c2),
                        X as X.c2 > prev(X.c2),
                        Y as Y.c2 < prev(Y.c2),
                        Z as Z.c2 > prev(Z.c2))
            as T
        ></query>
         <query id="stock"><![CDATA[
            select c1 as ts, c2 as price from StockInputsStream
        ></query>
    </rules>
</processor>
<channel>
    <name>StockOutputChannel</name>
    <selector>stock</selector>
</channel>
<channel>
    <name>MatchOutputChannel</name>
    <selector>match</selector>
</channel>
```

4.105 server-context-path

Use the `server-context-path` required element to specify the path to the server for every local http-pub-sub-adapter for publishing. The default value is `/pubsub`. If you have created a new local HTTP pub-sub server or changed the default configuration, then specify the appropriate `path` child element value.

Note:

Do not specify the `server-context-path` element for a remote HTTP pub-sub adapter.

This element has no child elements and no attributes.

The following example shows how to use the `server-context-path` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>localPub</name>
    <server-context-path>/pubsub</server-context-path>
    <channel>/test1</channel>
</http-pub-sub-adapter>
```

4.106 server-url

Use the `server-url` required element specifies the path to the server for every remote http-pub-sub-adapter for publishing or subscribing. The remote HTTP pub-sub server could be another instance of Oracle Event Processing, or a WebLogic Server instance, or it could be any third-party HTTP pub-sub server. For example:

```
http://myhost.com:9102/pubsub
```

Note:

Do not specify the `server-url` element option for a local HTTP pub-sub adapter.

This element has no child elements and no attributes.

The following example shows how to use the `server-url` element in the component configuration file. In the example, the URL of the remote HTTP pub-sub server to which the `remotePublisher` adapter will publish events is `http://myhost.com:9102/pubsub`.

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://myhost.com:9102/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
</http-pub-sub-adapter>
```

4.107 session

If more than one session are required, then group common configurations in the `default-session` element. All of the `session` tags by default inherit all of the elements declared in the `default-session` tag.

4.108 session-ack-mode-name

Use the `session-ack-mode-name` to define the session acknowledge mode name for a jms-adapter. This element has no child elements and no attributes.

Valid values from `javax.jms.Session` are the following. The default value is `AUTO_ACKNOWLEDGE`.

- `AUTO_ACKNOWLEDGE`: With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.
- `CLIENT_ACKNOWLEDG`: With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.
- `DUPS_OK_ACKNOWLEDGE`: This acknowledgment mode instructs the session to lazily acknowledge the delivery of messages.

The following example shows how to use the `session-ack-mode-name` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <session-ack-mode-name>AUTO_ACKNOWLEDGE</session-ack-mode-name>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.109 session-transacted

Use this element to define whether or not a session is transacted for both an inbound or outbound jms-adapter. This element has no child elements and no attributes.

Valid values are:

- `true`
- `false`

The following example shows how to use the `session-transacted` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <connection-jndi-name>weblogic.jms.ConnectionFactory</connection-jndi-name>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
    <session-ack-mode-name>AUTO_ACKNOWLEDGE</session-ack-mode-name>
```

```
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.110 source-url

Use the `source-url` entry to specify where a CSV Inbound adapter can find file that contains the inbound event data. This element has no child elements or attributes.

The following example shows the assembly file entry for StockTradeCSVInboundAdapter.

```
<wlevs:adapter id="StockTradeCSVInboundAdapter" provider="csv-inbound">
    <wlevs:listener ref="AdapterOutputChannel"/>
    <wlevs:instance-property name="eventType" value="TradeEvent"/>
    <wlevs:instance-property name="sourceUrl"  value="file:////scratch/mpawlan/
oep9-19/oep/utils/load-generator/StockData.csv"/>
</wlevs:adapter>
```

4.111 stage

Use the `stage` element to define the stage for a start-location or end-location element of a diagnostic profile. A valid value is the name of an existing stage in your Event Processing Network (EPN).

The following example shows how to use the `stage` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.112 start

Use the `start` element to define a start time for a time-range, time-range-offset, or schedule-time-range-offset element. This element has no child elements and no attributes.

Express the start time as an XML Schema `dateTime` value of the form:

```
yyyy-mm-ddThh:mm:ss
```

For example, to specify that play back should start on January 20, 2010, at 5:00am and end on January 20, 2010, at 6:00 pm, enter the following:

```
  <time-range>
    <start>2010-01-20T05:00:00</start>
    <end>2010-01-20T18:00:00</end>
  </time-range>
```

For complete details of the XML Schema `dateTime` format, see `http://www.w3.org/TR/xmlschema-2/#dateTime-lexical-representation`.

The following example shows how to use the `start` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range>
        <start>2010-01-20T05:00:00</start>
        <end>2010-01-20T18:00:00</end>
    </time-range>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.113 start-location

Use the `start-location` element to define the start location of a diagnostic profile. This element has the following child elements and no attributes.

- application
- stage
- direction

The following example shows how to use the `start-location` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
```

```
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.114 start-stage

Use the `start-stage` element to define the starting stage of a diagnostic profile. This element has no child elements and no attributes.

A valid value is the name of an existing stage in your Event Processing Network (EPN).

The following example shows how to use the `start-stage` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.115 threshold

Use the `threshold` element to define the threshold above which the Oracle Event Processing server logs a monitoring event. This element is applicable only in an `average-latency` element in a diagnostic profile.

This element has the following child elements and no attributes.

- amount
- unit

The following example shows how to use the `threshold` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <average-latency>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
```

```
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
            <threshhold>
                <amount>100</amount>
                <unit>MILLISECONDS</unit>
            </threshhold>
        </average-latency>
    </profile>
</diagnostic-profiles>
```

4.116 throughput

Use the `throughput` element to define a throughput diagnostic profile.

- name
- throughput-interval
- average-interval
- location

The following example shows how to use the `throughput` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <throughput>
            <throughput-interval>
                <amount>100000</amount>
                <unit>MICROSECONDS</unit>
            </throughput-interval>
            <average-interval>
                <amount>100000000</amount>
                <unit>NANOSECONDS</unit>
            </average-interval>
            <location>
                <application>diagnostic</application>
                <stage>AlertEventStream</stage>
                <direction>INBOUND</direction>
            </location>
        </throughput>
    </profile>
</diagnostic-profiles>
```

4.117 throughput-interval

Use the `throughput-interval` element to define the throughput interval of a diagnostic profile. This element has the following child elements and no attributes.

- amount
- unit

The following example shows how to use the `throughput-interval` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <throughput>
            <throughput-interval>
                <amount>100000</amount>
                <unit>MICROSECONDS</unit>
            </throughput-interval>
            <average-interval>
                <amount>100000000</amount>
                <unit>NANOSECONDS</unit>
            </average-interval>
            <location>
                <application>diagnostic</application>
                <stage>AlertEventStream</stage>
                <direction>INBOUND</direction>
            </location>
        </throughput>
    </profile>
</diagnostic-profiles>
```

4.118 time-to-live

Use the `time-to-live` element to define the maximum amount of time in milliseconds that an entry is cached. The default value is `infinite`. This element has no child elements and no attributes.

The following example shows how to use the `time-to-live` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.119 trace-parameters

Use the `trace-parameters` element to configure event tracing for a stage in the event processing network. This element has the following child elements and no attributes.

- channel-name
- active

The component configuration excerpt shown in the following example illustrates how you might configure a processor for event tracing. The trace-parameters element's active child element specifies that tracing is on, while the channel-name element specifies the HTTP pub-sub channel to which traced elements should be sent.

```
<processor>
    <name>FindCrossRates</name>
    <trace-parameters>
        <active>true</active>
        <channel-name>/NonClusteredServer/fx/FindCrossRates/output</channel-name>
    </trace-parameters>
    <rules>
        <!-- Query rules omitted. -->
    </rules>
</processor>
```

4.120 unit

Use the `unit` element to define the duration units for the amount element. This element has no child elements and no attributes.

Valid values are:

- `NANOSECONDS`
- `MICROSECONDS`
- `MILLISECONDS`
- `SECONDS`
- `MINUTES`
- `HOURS`
- `DAYS`

The following example shows how to use the `unit` element in the component configuration file:

```
<diagnostic-profiles>
    <name>myselfprofiles</name>
    <profile>
        <name>testProfile0</name>
        <enabled>true</enabled>
        <start-stage>MetricSubscriber</start-stage>
        <max-latency>
            <collect-interval>
                <amount>1000</amount>
                <unit>s</unit>
            </collect-interval>
            <name>testProfile0MaxLat</name>
            <start-location>
                <application>diagnostic</application>
                <stage>MetricSubscriber</stage>
                <direction>INBOUND</direction>
            </start-location>
            <end-location>
                <application>diagnostic</application>
                <stage>MonitorProcessor</stage>
                <direction>OUTBOUND</direction>
            </end-location>
        </max-latency>
    </profile>
</diagnostic-profiles>
```

4.121 user

Use the `user` element in the following parent elements:

- http-pub-sub-adapter: Use the `user` element to define the user name of the HTTP pub-sub server to which the Oracle Event Processing application is publishing requires user authentication.
- jms-adapter: When Oracle Event Processing acquires the JNDI `InitialContext`, it uses the user and password (or encrypted-password) elements. When Oracle Event Processing calls the `createConnection` method on the `javax.jms.ConnectionFactory` to create a connection to the JMS destination (JMS queue or topic), it uses the connection-user and connection-password (or connection-encrypted-password element), if configured. Otherwise, Oracle Event Processing uses the `user` and password (or encrypted-password) elements.
- rmi-adapter: Use the `user` element for RMI authentication to the Oracle WebLogic Server.
- edn-adapter: Use the `user` element for Oracle SOA Suite EDN or JMS authentication.

This element has no child elements and no attributes.

The following example shows how to use the `user` element in the component configuration file:

```
<http-pub-sub-adapter>
    <name>remotePub</name>
    <server-url>http://localhost:9002/pubsub</server-url>
    <channel>/test1</channel>
    <event-type>com.bea.wlevs.tests.httppubsub.PubsubTestEvent</event-type>
    <user>wlevs</user>
    <password>wlevs</password>
</http-pub-sub-adapter>
```

4.122 validate

Use the `validate` element to perform schema validation during JAXB marshalling and unmarshalling. Set to `true` to enable validation, and set to `false` to disable validation.

The `validate` element has no child elements and no attributes.

The following example shows how to use the validate element in a configuration file.

```
<edn-adapter>
  <name>ednInboundAdapterJaxb</name>
  <edl-file>CallCenterEvents.edl</edl-file>
  <schema-file>callcenter.xsd</schema-file>
  <validate>true</validate>
  <package-name>org.callcenter</package-name>
  <raw-xml-content>false</raw-xml-content>
  <jndi-provider-url>vm://localhost</jndi-provider-url>
  <jndi-factory>org.apache.activemq.jndi.ActiveMQInitialContextFactory
    </jndi-factory>
  <user>test1</user>
  <password>test123</password>
</edn-adapter>
```

4.123 value

Use the `value` element to define the value of a parameter element. This element has no child elements and no attributes.

The following example shows how to use the `value` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
       <store-policy-parameters>
           <parameter>
               <name>timeout</name>
               <value>300</value>
           <parameter>
       </store-policy-parameters>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

4.124 view

Use the `view` element to define an Oracle CQL view for a component. This element has no child elements and the following attributes.

Table 4-3 Attributes of the view Component Configuration Element

Attribute	Description	Data Type	Required ?
`id`	Unique identifier for this query.	`String`	Yes.
`active`	Execute this query when the application is deployed and run. Valid values are `true` and `false`. Default value is `false`.	`Boolean`	No.
`ordering-constraint`	Enable or disable parallel query execution, through which events can be processed in parallel rather than serially. The attribute supports one of the following three values: • `ORDERED` means that the query must handle events serially. This is the default behavior. • `UNORDERED` means that, whenever possible, the CQL processor executes in parallel on multiple threads to process the events. • `PARTITION_ORDERED` means that when the query is partitioning events, ensure total order within a partition and (if possible) disregard order across partitions.	String	No.
`partition-expression`	The partition expression (used in the CQL code) that should be the basis for relaxing the cross-partition ordering constraint.	String	No.

Table 4-3 (Cont.) Attributes of the view Component Configuration Element

Attribute	Description	Data Type	Required ?
`schema`	Space delimited list of stream elements used in the view.	`String` of space delimited tokens.	No.

The following example shows how to use the `view` element in the component configuration file:

```
<processor>
    <name>cqlProcessor</name>
    <rules>
        <view id="lastEvents" schema="cusip bid srcId bidQty ask askQty seq"><![CDATA[
            select cusip, bid, srcId, bidQty, ask, askQty, seq
            from inputChannel[partition by srcId, cusip rows 1]
        ></view>
        <view id="bidask" schema="cusip bid ask"><![CDATA[
            select cusip, max(bid), min(ask)
            from lastEvents
            group by cusip
        ></view>
            <view ...><![CDATA[
                ...
        ></view>
        ...
        <view id="MAXBIDMINASK" schema="cusip bidseq bidSrcId bid askseq askSrcId ask bidQty
askQty"><![CDATA[
            select bid.cusip, bid.seq, bid.srcId as bidSrcId, bid.bid, ask.seq,
                ask.srcId as askSrcId, ask.ask, bid.bidQty, ask.askQty
            from BIDMAX as bid, ASKMIN as ask
            where bid.cusip = ask.cusip
        ></view>
        <query id="BBAQuery"><![CDATA[
            ISTREAM(select bba.cusip, bba.bidseq, bba.bidSrcId, bba.bid, bba.askseq,
                bba.askSrcId, bba.ask, bba.bidQty, bba.askQty,
                "BBAStrategy" as intermediateStrategy, p.seq as correlationId, 1 as priority
            from MAXBIDMINASK as bba, inputChannel[rows 1] as p where bba.cusip = p.cusip)
        ></query>
    </rules>
</processor>
```

4.125 work-manager

Use the `work-manager` element to define the name of a work manager for a jms-adapter. The element has no child elements and no attributes.

Valid value is the name specified in the Oracle Event Processing server file `work-manager` element `name` child element. The default value is the work manager configured for the application itself.

The following example shows how to use the `work-manager` element in the component configuration file:

```
<jms-adapter>
    <name>jmsInbound-text</name>
    <jndi-provider-url>t3://localhost:7001</jndi-provider-url>
    <destination-name>JMSServer-0/Module1!Queue1</destination-name>
    <user>weblogic</user>
    <password>weblogic</password>
    <work-manager>JettyWorkManager</work-manager>
```

```
    <concurrent-consumers>1</concurrent-consumers>
    <session-transacted>false</session-transacted>
</jms-adapter>
```

4.126 work-manager-name

Use the work-manager-name element to define a work manager for a cache. This element has no child elements and no attributes.

The listeners element has a single child element, work-manager-name, that specifies the work manager to be used for asynchronously invoking listeners. This value is ignored if synchronous invocations are enabled. If a work manager is specified for the cache itself, this value overrides it for invoking listeners only.

Valid value is the name specified in the Oracle Event Processing server file `work-manager` element `name` child element. The default value is the work manager configured for the application itself.

The following example shows how to use the `work-manager-name` element in the component configuration file:

```
<cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
</cache>
```

4.127 write-behind

Use the `write-behind` element to specify asynchronous writes to the cache store. The cache store is invoked from a separate thread after a create or update of a cache entry. This element may be changed dynamically.

This element has the following child elements and no attributes.

- work-manager-name
- batch-size
- buffer-size
- buffer-write-attempts
- buffer-write-timeout

The following example shows how to use the `write-behind` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-behind>
```

```
            <work-manager-name>JettyWorkManager</work-manager-name>
            <batch-size>100</batch-size>
            <buffer-size>100</buffer-size>
            <buffer-write-attempts>100</buffer-write-attempts>
            <buffer-write-timeout>100</buffer-write-timeout>
        </write-behind>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.128 write-none

Use the `write-none` element to specify no writes to a cache store. No writes is the default write policy. You can change this element dynamically. This element has no child elements and no attributes.

The following example shows how to use the `write-none` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-none/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

4.129 write-through

Use the write-through element to specify synchronous writes to the cache store. As soon as an entry is created or updated the write occurs. You can change this element dynamically.

This element has no child elements and no attributes.

The following example shows how to use the `write-through` element in the component configuration file:

```
<caching-system>
    <name>providerCachingSystem</name>
    <cache>
        <name>providerCache</name>
        <max-size>1000</max-size>
        <eviction-policy>FIFO</eviction-policy>
        <time-to-live>60000</time-to-live>
        <idle-time>120000</idle-time>
        <write-through/>
        <work-manager-name>JettyWorkManager</work-manager-name>
        <listeners asynchronous="false">
            <work-manager-name>JettyWorkManager</work-manager-name>
        </listeners>
    </cache>
</caching-system>
```

5

Event Record and Playback Schema

This chapter provides a reference to the elements of the `wlevs_eventstore_config.xsd` schema. This schema is behind the XML files you use to configure Oracle Event Processing event record and play back.

This chapter includes the following sections:

5.1 batch-size

Use this element to define the number of updates that are picked up from the store buffer to write back to the backing store. This element can be changed dynamically. The batch-size element has no child elements and no attributes.

The following example shows how to use the `batch-size` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
```

```
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.2 batch-time-out

Use this element to define The number of seconds event buffer will wait to accumulate batch-size number of events before to write to the event store. This element has no child elements and no attributes.

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

The following example shows how to use the `batch-time-out` element in the component configuration file:

5.3 dataset-name

Use the `dataset-name` element to define the group of data that the user wants to group together. In the case of the Oracle RDBMS-based provider, it specifies the database area, or schema, in which the tables that store the recorded events are created. When configuring the Oracle RDBMS-based provider, you are required to specify this element.

This element has no child elements and no attributes.

The following example shows how to use the `dataset-name` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.4 event-type-list

Use the `event-type-list` element to define one or more events for record or playback for a component. This element has the event-type child element and no attributes.

The following example shows how to use the `event-type-list` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
```

```
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.5 playback-speed

Use the `playback-speed` element to define the playback speed as a positive float. The default value is 1, which corresponds to normal speed. A value of 2 means that events will be played back 2 times faster than the original record speed. Similarly, a value of 0.5 means that events will be played back at half the speed.

This element has no child elements an no attributes.

The following example shows how to use the `duration` element in the component configuration file:

```
<playback-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range-offset>
        <start>2010-01-20T05:00:00</start>
        <duration>03:00:00</duration>
    </time-range-offset>
    <playback-speed>100</playback-speed>
</playback-parameters>
```

5.6 provider-name

Use the `provider-name` element in the following parent elements:

- netio: Use the `provider-name` element to define which provider to use for the underlying socket implementation. Valid value is an Oracle Event Processing server file `netio` child element `provider-type`.
- record-parameters: Use the `provider-name` element to define the name of the event store provider. The value of this element corresponds to the value of the `name` child element of the `rdbms-event-store-provider` element in the file of the Oracle Event Processing server instance.

  When configuring the Oracle RDBMS-based provider, you are required to specify this element. When the `provider-name` element is blank, the default Berkeley database provider is used.

This element has no child elements and no attributes.

The following example shows how to use the `provider-name` element in the component configuration file:

```
<netio>
        <provider-name>providerCache</provider-name>
        <num-threads>1000</num-threads>
</netio>
```

5.7 recording-session-name

Use the `recording-session-name` element to specify a name for the recording session so that you can access the session programmatically. This element has no child elements and no attributes.

The following example shows how to use this element in a configuration file.

```
 <record-parameters>
    <dataset-name>tuple1</dataset-name>
    <recording-session-name>sessionname</recording-session-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        </parameter>
    </store-policy-parameters>
    <schedule-time-range>
        <start>2010-01-20T05:00:00</start>
        <end>2010-01-20T18:00:00</end>
    </schedule-time-range>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.8 schedule-time-range

Use the `schedule-time-range` element to define the time during which events will be played back to the stage. Playing back starts at the specified start time and continues until all of the events are played back or until the specified end time. When repeat is set to `true`, playback continues until the specified end time or until playback is explicitly stopped by the user. This element applies only to the playback-parameters element.

This element has the following child elements and no attributes.

- start
- end

The following example shows how to use the `schedule-time-range` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        </parameter>
    </store-policy-parameters>
    <schedule-time-range>
        <start>2010-01-20T05:00:00</start>
        <end>2010-01-20T18:00:00</end>
```

```
    </schedule-time-range>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.9 schedule-time-range-offset

Use this element to define the time during which events will be played back to the stage. Playing back will start at the specified start time and will continue until all the events are played back or specified end time. If repeat is set to `true`, playback will continue until the specified end time or until playback is explicitly stopped by the user. This element applies only to the playback-parameters element.

This element has the following child elements and no attributes.

- start
- duration

The following example shows how to use the `schedule-time-range-offset` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <schedule-time-range-offset>
        <start>2010-01-20T05:00:00</start>
        <duration>03:00:00</duration>
    </schedule-time-range-offset>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.10 store-policy-parameters

Use the `store-policy-parameters` element to define one or more store policy parameters that are specific to the event store provider. This element has the parameter element and no attributes.

The following example shows how to use the `store-policy-parameter` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
        <event-type-list>
            <event-type>TupleEvent1</event-type>
        </event-type-list>
        <provider-name>test-rdbms-provider</provider-name>
       <store-policy-parameters>
           <parameter>
               <name>timeout</name>
               <value>300</value>
           <parameter>
       </store-policy-parameters>
    <batch-size>1</batch-size>
```

```
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.11 time-range

Use the `time-range` element to define a filter that the Oracle Event Processing server applies to the events in the event store. Only events with a `record-time` in this time range are played back to the stage.

Note:

Use either time-range-offset or time-range, but not both.

This element has the following child elements and no attributes.

- start
- end

The following example shows how to use the `time-range` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range>
        <start>2010-01-20T05:00:00</start>
        <end>2010-01-20T18:00:00</end>
    </time-range>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

5.12 time-range-offset

Use the `time-range-offset` element to define a filter that the Oracle Event Processing server applies to the events in the event store. Only events with a `record-time` in this time range are played back to the stage.

Note:

Use either time-range or time-range-offset, but not both.

This element has the following child elements and no attributes.

- start
- duration

The following example shows how to use the `time-range-offset` element in the component configuration file:

```
<record-parameters>
    <dataset-name>tuple1</dataset-name>
    <event-type-list>
        <event-type>TupleEvent1</event-type>
    </event-type-list>
    <provider-name>test-rdbms-provider</provider-name>
    <store-policy-parameters>
        <parameter>
            <name>timeout</name>
            <value>300</value>
        <parameter>
    </store-policy-parameters>
    <time-range-offset>
        <start>2010-01-20T05:00:00</start>
        <duration>03:00:00</duration>
    </time-range-offset>
    <batch-size>1</batch-size>
    <batch-time-out>10</batch-time-out>
</record-parameters>
```

6

Deployment Schema

This chapter provides a reference to the elements of the `deployment.xsd` schema. The `deployment.xsd` schema is behind the XML with which you configure Oracle Event Processing application deployment.

This chapter includes the following sections:

6.1 Deployment Elements and Hierarchy

Oracle Event Processing provides a number of application assembly elements that you use in the EPN assembly file of your application to register event types, declare the components of the event processing network and specify how they are linked together. The EPN assembly file is an extension to the standard Spring context file.

The Oracle Event Processing component configuration elements are organized into the following hierarchy:

```
beans
    Standard Spring and OSGi elements such as bean, osgi-service, and so on.
```

6.2 wlevs:deployment

Use this element to declare an adapter component to the Spring application context. This element has no child elements and the following attributes.

Table 6-1 Attributes of the wlevs:deployment Deployment Element

Attribute	Description	Data Type	Required?
`id`	Unique identifier for this deployed application.	`String`	Yes.
`depends-on`	The names of the beans that this deployment bean depends on being initialized. The bean factory will guarantee that these beans get initialized before this bean.	`String`	No.

Table 6-1 (Cont.) Attributes of the wlevs:deployment Deployment Element

Attribute	Description	Data Type	Required?
`location`	URL that specifies the location of the bundle that is to be deployed. If a relative URL is specified then the location is relative the `DOMAIN_DIR` domain directory. For example: `location="file:applications/simpleApp/simpleApp.jar"` Specifies that the bundle `simpleApp.jar`, located in the *`DOMAIN_DIR`*`/applications/simpleApp` directory, is to be deployed to Oracle Event Processing server.	`String`	No.
`state`	Specifies the state that the bundle should be in once it is deployed to the Oracle Event Processing server. The value of this attribute must be one of the following: • `start`: Install and start the bundle so that it immediately begins taking client requests. • `install`: Install the bundle, but do not start it. • `update`: Update an existing bundle. Default value: `start`.	`TState`	No.
type	Specifies the type of a bundle. The value of this attribute must be one of the following: • `library`: This is a library bundle. • `application`: This is an Oracle Event Processing application. Default value: `application`.	TBundleType	No
group-name	Specifies the name of the cluster group to which the application is deployed.	String	No
start-level		int	No
apply-parameters	Specifies if application parameters related to an application meta-type resource should be applied. The default behavior is to instantiate only once the first time the application is deployed.	TapplyParams	No

6.2.1 Example

This is how to use the `wlevs:deployment` element in the deployment file:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wlevs="http://www.bea.com/ns/wlevs/deployment" xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
```

```
    http://www.bea.com/ns/wlevs/deployment
    http://www.bea.com/ns/wlevs/deployment/deployment.xsd">
    <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
<property name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE"/>
</bean>
<wlevs:deployment
    id="fx"
    state="start"
    location="file:${wlevs.domain.home}/applications/fx/com.bea.wlevs.example.fx_11.1.0.0.jar"/>
</beans>
```

7

Server Configuration Schema

This chapter provides a reference to elements of the `wlevs_server_config.xsd` schema, the schema behind XML you use to configure Oracle Event Processing server attributes and services such as logging, Oracle Continuous Query Language (CQL), Secure Sockets Layer (SSL), Java Management Extensions (JMX), HTTP Publish-Subscribe, and more.

This chapter includes the following sections:

- jetty-web-app
- jmx
- jndi-context
- log-file
- log-stdout
- logging-service
- message-filters
- name
- netio
- netio-client
- partition-order-capacity
- path
- pubsub-bean
- rdbms-event-store-provider
- rmi
- scheduler
- server-config
- services
- show-detail-error-message
- ssl
- timeout-seconds
- transaction-manager
- use-secure-connections
- weblogic-instances
- weblogic-jta-gateway
- weblogic-rmi-client
- work-manager
- xa-params

7.1 Oracle Event Processing Server Configuration Elements

Oracle Event Processing provides a number of server configuration elements that you use to configure Oracle Event Processing server-specific attributes and services.

The top-level Oracle Event Processing server configuration elements are organized into the following hierarchy:

```
config
  domain
  rmi
  jndi-context
  exported-jndi-context
  jmx
  transaction-manager
  work-manager
  logging-service
  log-stdout
  log-file
  jetty-web-app
  netio
  jetty
  glassfish-ws
  netio-client
  debug
  data-source
  http-pubsub
  event-store
  cluster
  bdb-config
  rdbms-event-store-provider
  user-event-store-provider
  ssl
  weblogic-rmi-client
  weblogic-jta-gateway
  use-secure-connections
  show-detail-error-message
  cql
  event-inspector
```

7.2 auth-constraint

Use the `auth-constraint` element to configure an authorization constraint for a channel-constraints element. This element has the following child elements and no attributes.

Table 7-1 Child Elements of: auth-constraint

XML Tag	Type	Description
`description`	`string`	The description of the role.
`role-name`	`string`	A valid role name.

For more information on channels, see channels.

The following example shows how to use the `auth-constraint` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
...
        <channel-constraints>
```

```
            <element>
...
                <auth-constraint>
                    <description>Administrators</description>
                    <role-name>admin</role-name>
                </auth-constraint>
            </element>
        </channel-constraints>
    </pub-sub-bean>
</http-pubsub>
```

7.3 bdb-config

Use the `bdb-config` element to configure the default event store provider that uses a Berkeley database instance.

Optionally, you can configure the Oracle Event Processing server to use a relational database instance as the event store provider as rdbms-event-store-provider describes.

This element has the following child elements and no attributes.

Table 7-2 Child Elements of: bdb-config

XML Tag	Type	Description
`db-env-path`	`string`	Specifies the subdirectory in which Oracle Event Processing server creates Berkeley database instances relative to the *`DOMAIN_DIR`*`/`*`servername`*`/config` directory of your server, where *`DOMAIN_DIR`* refers to the domain directory, such as `/oracle_cep/user_projects/domains/myDomain` and *`servername`* refers to the name of your server, such as `defaultserver`. Default: `bdb`
`cache-size`	`long`	Specifies the amount of memory, in bytes, available for Berkeley database cache entries. You can adjust the cache size to tune Berkeley database performance. Default: `je.maxMemoryPercent * JVM maximum memory`

The following example shows how to use the `bdb-config` element in the Oracle Event Processing server configuration file:

```
<bdb-config>
    <db-env-path>bdb</db-env-path>
    <cache-size>1000</cache-size>
</bdb-config>
```

7.4 calendar

Use the `calendar` element to configure cql calendar options in the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-3 Child Elements of: calendar

XML Tag	Type	Description
`date-format`	`string`	String must match the description of `java.text.DateFormat`.
`timezone`	`string`	String must match the description of `java.util.TimeZone`.

The following example shows how to use the `calendar` element in the Oracle Event Processing server configuration file. In the example, the `cql` element's unique identifier is `myCQL`.

```
<cql>
    <name>myCQL</name>
    <storage>
        <folder>myfolder</folder>
        <metadata-name>myname</metadata-name>
    </storage>
    <calendar>
        <date-format>myclass</date-format>
        <timezone>10</timezone>
    </calendar>
    <scheduler>
        <class-name>myclass</class-name>
        <threads>10</threads>
        <direct-interop>false</direct-interop>
    </scheduler>
</cql>
```

7.5 channels

Use the `channels` element to configure one or more channels for a pubsub-bean element. Channel patterns always begin with a forward slash (`/`). Clients subscribe to these channels to either publish or receive messages.

This element has one or more `element` child elements that each contain a `channel-pattern` child element and zero or more `message-filters` child elements. Each `message-filters` child element contains an `element` child element with the `string` value of a `message-filter-name` that corresponds to a message-filters element.

This element has no attributes.

The following example shows how to use the `channels` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
        <server-config>
            <supported-transport>
                <types>
                    <element>long-polling</element>
                </types>
            </supported-transport>
            <publish-without-connect-allowed>
                true
```

```
            </publish-without-connect-allowed>
        </server-config>
        <channels>
            <element>
                <channel-pattern>/evsmonitor</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsalert</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsdomainchange</channel-pattern>
            </element>
        </channels>
    </pub-sub-bean>
</http-pubsub>
```

7.6 channel-constraints

Use the `channel-constraints` element to configure one or more channel constraints for a pubsub-bean element. This element has the following child elements and no attributes.

- channel-resource-collection
- auth-constraint

For more information on channels, see channels.

The following example shows how to use the `channel-constraints` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
...
        <channel-constraints>
            <element>
                <channel-resource-collection>
                    <element>
                        <channel-resource-name>Foo</channel-resource-name>
                        <descriptions>
                            <element>Foo</element>
                        </descriptions>
                        <channel-patterns>
                            <element>Foo</element>
                        </channel-patterns>
                        <channel-operations>
                            <element>Foo</element>
                        </channel-operations>
                    </element>
                </channel-resource-collection>
                <auth-constraint>
                    <description>Foo</description>
                    <role-name>Foo</role-name>
                </auth-constraint>
            </element>
        </channel-constraints>
    </pub-sub-bean>
</http-pubsub>
```

7.7 channel-resource-collection

Use the `channel-resource-collection` element to configure one or more channel resource collections for a channel-constraints element. This element has the following child elements and no attributes.

Table 7-4 Child Elements of: channel-resource-collection

XML Tag	Type	Description
`channel-resource-name`	`string`	The name of this channel resource.
`descriptions`	`string`	Description of this channel resource collection. This element contains an `element` child element with a `string` value.
`channel-patterns`	`string`	Specifies a channel pattern. This element contains an `element` child element with a `string` value.
`channel-operations`	`string`	Specifies the operation to channel, validate values include: • `create` • `delete` • `subscribe` • `publish` This element contains an `element` child element with a `string` value.

For more information on channels, see channels.

The following example shows how to use the `channel-resource-collection` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
...
        <channel-constraints>
            <element>
                <channel-resource-collection>
                    <element>
                        <channel-resource-name>Foo</channel-resource-name>
                        <descriptions>
                            <element>Foo</element>
                        </descriptions>
                        <channel-patterns>
                            <element>Foo</element>
                        </channel-patterns>
                        <channel-operations>
                            <element>Foo</element>
                        </channel-operations>
                    </element>
                </channel-resource-collection>
                <auth-constraint>
                    <description>Foo</description>
```

```
                    <role-name>Foo</role-name>
                </auth-constraint>
            </element>
        </channel-constraints>
    </pub-sub-bean>
</http-pubsub>
```

7.8 cluster

Use the `cluster` element to configure a cluster component in the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-5 Child Elements of: cluster

XML Tag	Type	Description
`name`	`string`	The name of this cluster. For more information, see name.
`server-name`	`string`	Specifies a unique name for the server. Oracle Event Processing Visualizer uses the value of this element when it displays the server in its console. Default value: • Oracle Event Processing native clustering: `WLEvServer-`*`identity`* where *`identity`* is the value of the `identity` element. • Oracle Coherence: `WLEvServer-`*`identity`* where *`identity`* is the member ID as determined by Oracle Coherence.
`server-host-name`	`string`	Specifies the host address or IP used for point-to-point HTTP multiserver communication. Default value is the IP address associated with the default NIC for the machine.
`multicast-address`	`string`	This child element is required unless all servers of the multiserver domain are hosted on the same computer; in that case you can omit the `multicast-address` element and Oracle Event Processing automatically assigns a multicast address to the multiserver domain based on the computer's IP address. If, however, the servers are hosted on different computers, then you must provide an appropriate domain-local address. Oracle recommends you use an address of the form *`239.255.X.X`*, which is what the auto-assigned multicast address is based on. All the Oracle Event Processing servers using this `multicast-address` must be on the same subnet. Using Oracle Coherence, there is also an extension: if you use a unicast address then Oracle Coherence will be configured in Well Known Address (WKA) mode. This is necessary in environments that do not support multicast.

Table 7-5 (Cont.) Child Elements of: cluster

XML Tag	Type	Description
`multicast-interface`	`string`	The name of the interface that the multicast address should be bound to. This can be one of: • Simple name, such as `eth0`. • IP address to which the NIC is bound, such as `192.168.1.2`. • IP address and network mask to which the NIC is bound separated by a `/`, such as `192.68.1.2/255.255.255.0`.
`multicast-port`	`int`	Specifies the port used for multicast traffic. Default value is `9100`.
`identity`	`string`	Applicable only to Oracle Event Processing native clustering: specifies the server's identity and must be an integer between 1 and `INT_MAX`. Oracle Event Processing numerically compares the server identities during multiserver operations; the server with the lowest identity becomes the domain coordinator. Be sure that each server in the multiserver domain has a different identity; if servers have the same identity, the results of multiserver operations are unpredictable. Not applicable to Oracle Coherence.
`enabled`	See Description	Specifies whether or not the cluster is enabled. Valid values: • `coherence` • `evs4j` • `true`: cluster is enabled (Oracle Coherence mode) • `false`: cluster is not enabled (default).
`security`	See Description	Specifies the type of security for this cluster. Valid values: • `none`—Default value. Specifies that no security is configured for the multiserver domain. • `encrypt`—Specifies that multiserver messages should be encrypted.
`groups`	`string`	Specifies a comma-separated list of the names of the groups this cluster belongs to.
`operation-timeout`	`int`	Specifies, in milliseconds, the time out for point-to-point HTTP multiserver requests. Default value is `30000`.

The following example shows how to use the `cluster` element in the Oracle Event Processing server configuration file:

```
<cluster>
    <name>MyCluster</name>
    <server-name>myServer1</server-name>
    <multicast-address>239.255.0.1</multicast-address>
    <identity>1</identity>
    <enabled>true</enabled>
</cluster>
```

7.9 connection-pool-params

Use the `connection-pool-params` element to specify connection pool-related data-source parameters. This element has the following child elements and no attributes.

Table 7-6 Child Elements of: connection-pool-params

XML Tag	Type	Description
`statement-timeout`	`int`	The time after which a statement currently being executed will time out. `statement-timeout` relies on underlying JDBC driver support. The server passes the time specified to the JDBC driver using the `java.sql.Statement.setQueryTimeout` method. If your JDBC driver does not support this method, it may throw an exception and the timeout value is ignored. A value of -1 disables this feature. A value of 0 means that statements will not time out. Default: -1.
`profile-harvest-frequency-seconds`	`int`	The number of seconds between diagnostic profile harvest operations. Default: 300.
`inactive-connection-timeout-seconds`	`int`	The number of inactive seconds on a reserved connection before the connection is reclaimed and released back into the connection pool. Default: 0.
`shrink-frequency-seconds`	`int`	The number of seconds to wait before shrinking a connection pool that has incrementally increased to meet demand. Default: 900.
`driver-interceptor`	`string`	Specifies the absolute name of the application class used to intercept method calls to the JDBC driver. The application specified must implement the `weblogic.jdbc.extensions.DriverInterceptor` interface.
`seconds-to-trust-an-idle-pool-connection`	`int`	The number of seconds within a connection use that the server trusts that the connection is still viable and will skip the connection test, either before delivering it to an application or during the periodic connection testing process. Default: 10.
`pinned-to-thread`	`boolean`	This option can improve performance by enabling execute threads to keep a pooled database connection even after the application closes the logical connection. Default: `false`.
`test-connections-on-reserve`	`boolean`	Test a connection before giving it to a client. Requires that you specify `test-table-name`. Default: `false`.
`profile-type`	`int`	Specifies that type of profile data to be collected.

Table 7-6 (Cont.) Child Elements of: connection-pool-params

XML Tag	Type	Description
`statement-cache-type`	`string`	The algorithm used for maintaining the prepared statements stored in the statement cache. Valid values: • `LRU` - when a new prepared or callable statement is used, the least recently used statement is replaced in the cache • `FIXED` - the first fixed number of prepared and callable statements are cached Default: `LRU`.
`connection-reserve-timeout-seconds`	`int`	The number of seconds after which a call to reserve a connection from the connection pool will timeout. When set to 0, a call will never timeout. When set to -1, a call will timeout immediately. Default: -1.
`credential-mapping-enabled`	`boolean`	Enables the server to set a light-weight client ID on the database connection based on a map of database IDs when an application requests a database connection. Default: `false`.
`login-delay-seconds`	`int`	The number of seconds to delay before creating each physical database connection. This delay supports database servers that cannot handle multiple connection requests in rapid succession. The delay takes place both during initial data source creation and during the lifetime of the data source whenever a physical database connection is created. Default: 0.
`test-table-name`	`string`	The name of the database table to use when testing physical database connections. This name is required when you specify `test-frequency-seconds` and enable `test-reserved-connections`. The default SQL code used to test a connection is `select count(*) from` *`test-table-name`* where *`test-table-name`* is the value of the test-table-name element. Most database servers optimize this SQL to avoid a table scan, but it is still a good idea to set `test-table-name` to the name of a table that is known to have few rows, or even no rows. If test-table-name begins with `SQL`, then the rest of then the rest of the string following that leading token will be taken as a literal SQL statement that will be used to test connections instead of the standard query.
`statement-cache-size`	`int`	The number of prepared and callable statements stored in the cache between 1 and 1024. This may increase server performance. Default: 10.
`init-sql`	`string`	SQL statement to execute that will initialize newly created physical database connections. Start the statement with `SQL` followed by a space.

Table 7-6 (Cont.) Child Elements of: connection-pool-params

XML Tag	Type	Description
`connection-creation-retry-frequency-seconds`	`int`	The number of seconds between attempts to establish connections to the database. If you do not set this value, data source creation fails if the database is unavailable. If set and if the database is unavailable when the data source is created, the server will attempt to create connections in the pool again after the number of seconds you specify, and will continue to attempt to create the connections until it succeeds. When set to 0, connection retry is disabled. Default: 0.
`test-frequency-seconds`	`int`	The number of seconds between when the server tests unused connections. (Requires that you specify a Test Table Name.) Connections that fail the test are closed and reopened to re-establish a valid physical connection. If the test fails again, the connection is closed. In the context of multi data sources, this attribute controls the frequency at which the server checks the health of data sources it had previously marked as unhealthy. When set to 0, the feature is disabled. Default: 120.
`jdbc-xa-debug-level`	`int`	Specifies the JDBC debug level for XA drivers. Default: 10.
`initial-capacity`	`int`	The number of physical connections to create when creating the connection pool in the data source. If unable to create this number of connections, creation of the data source will fail. Default: 1.
`max-capacity`	`int`	The maximum number of physical connections that this connection pool can contain. Default: 15.
`capacity-increment`	`int`	The number of connections created when new connections are added to the connection pool. Default: 1.
`highest-num-waiters`	`int`	The maximum number of connection requests that can concurrently block threads while waiting to reserve a connection from the data source's connection pool. Default: `Integer.MAX_VALUE`.

The following example shows how to use the `connection-pool-params` element in the Oracle Event Processing server configuration file:

```
<data-source>
    <name>orads</name>
    <xa-params>
        <keep-xa-conn-till-tx-complete>true</keep-xa-conn-till-tx-complete>
    </xa-params>
    <driver-params>
        <url>jdbc:oracle:thin:@localhost:1521:ce102</url>
        <driver-name>oracle.jdbc.OracleDriver</driver-name>
        <properties>
            <element>
                <name>user</name>
                <value>wlevs</value>
```

```
            </element>
            <element>
                <name>password</name>
                <value>wlevs</value>
            </element>
        </properties>
    </driver-params>
    <connection-pool-params>
        <initial-capacity>5</initial-capacity>
        <max-capacity>10</max-capacity>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <test-frequency-seconds>5</test-frequency-seconds>
    </connection-pool-params>
    <data-source-params>
        <jndi-names>
            <element>orads</element>
        </jndi-names>
        <global-transactions-protocol>None</global-transactions-protocol>
    </data-source-params>
</data-source>
```

7.10 cql

Use the `cql` element to configure Oracle CQL-specific options in the Oracle Event Processing server. This element has the following child elements and no attributes.

- name
- calendar
- scheduler
- calendar
- partition-order-capacity

The following example shows how to use the `cql` element in the Oracle Event Processing server configuration file:

```
<cql>
    <name>myCQL</name>
    <storage>
        <folder>myfolder</folder>
        <metadata-name>myname</metadata-name>
    </storage>
    <scheduler>
        <class-name>myclass</class-name>
        <threads>10</threads>
        <direct-interop>false</direct-interop>
    </scheduler>
</cql>
```

7.11 data-source

This `data-source` element defines configuration for a data source service.

- name
- xa-params
- data-source-params
- connection-pool-params

- driver-params

The following example shows how to use the `data-source` element in the Oracle Event Processing server configuration file. In the example, the `data-source` element's unique identifier is `orads`.

```
<data-source>
    <name>orads</name>
    <driver-params>
        <url>jdbc:oracle:thin:@localhost:1521:ce102</url>
        <driver-name>oracle.jdbc.OracleDriver</driver-name>
        <properties>
            <element>
                <name>user</name>
                <value>wlevs</value>
            </element>
            <element>
                <name>password</name>
                <value>wlevs</value>
            </element>
        </properties>
    </driver-params>
    <connection-pool-params>
        <initial-capacity>5</initial-capacity>
        <max-capacity>10</max-capacity>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <test-frequency-seconds>5</test-frequency-seconds>
    </connection-pool-params>
    <data-source-params>
        <jndi-names>
            <element>orads</element>
        </jndi-names>
        <global-transactions-protocol>None</global-transactions-protocol>
    </data-source-params>
</data-source>
```

7.12 data-source-params

Use the `data-source-params` element to specify data source-related data-source parameters. This element has the following child elements and no attributes.

Table 7-7 Child Elements of: data-source-params

XML Tag	Type	Description
`algorithm-type`	See Description	The algorithm determines the connection request processing for the multi data source. Valid values: • `Failover` • `Load-Balancing` Default: `Failover`.
`stream-chunk-size`	`int`	Specifies the data chunk size for steaming data types between 1 and 65536. Default: 256.
`row-prefetch`	`boolean`	Specifies whether or not multiple rows to be prefetched (that is, sent from the server to the client) in one server access. Default: `false`.
`data-source-list`	`string`	The list of data sources to which the multi data source will route connection requests. The order of data sources in the list determines the failover order.

Table 7-7 (Cont.) Child Elements of: data-source-params

XML Tag	Type	Description
`failover-request-if-busy`	`boolean`	For multi data sources with the Failover algorithm, enables the multi data source to failover connection requests to the next data source if all connections in the current data source are in use. Default: `false`.
`row-prefetch-size`	`int`	If row prefetching is enabled, specifies the number of result set rows to prefetch for a client between 2 and 65536. Default: 48.
`jndi-names`	See Description	The JNDI path to where this Data Source is bound. By default, the JNDI name is the name of the data source. This element contains the following child elements: • `element`: contains the string name of a valid data-source element. For more information, see data-source. • `config-data-source-DataSourceParams-JNDINames`.
`scope`	`boolean`	Specifies the scoping of the data source. Note that `Global` is the only scoped supported by MSA. Default: `Global`.
`connection-pool-failover-callback-handler`	`string`	The name of the application class to handle the callback sent when a multi data source is ready to failover or fail back connection requests to another data source within the multi data source. The name must be the absolute name of an application class that implements the `weblogic.jdbc.extensions.ConnectionPoolFailoverCallback` interface.
`global-transactions-protocol`	`int`	Determines the transaction protocol (global transaction processing behavior) for the data source. Valid values: • `TwoPhaseCommit` - Standard XA transaction processing. Requires an XA driver • `LoggingLastResource` - A performance enhancement for one non-XA resource • `EmulateTwoPhaseCommit` - Enables one non-XA resource to participate in a global transaction, but has some risk to data • `OnePhaseCommit` - One-phase XA transaction processing using a non-XA driver. This is the default setting • `None` - Support for local transactions only Default: `OnePhaseCommit`.

The following example shows how to use the `data-source-params` element in the Oracle Event Processing server configuration file:

```
<data-source>
    <name>orads</name>
    <xa-params>
        <keep-xa-conn-till-tx-complete>true</keep-xa-conn-till-tx-complete>
    </xa-params>
```

```
    <driver-params>
        <url>jdbc:oracle:thin:@localhost:1521:ce102</url>
        <driver-name>oracle.jdbc.OracleDriver</driver-name>
        <properties>
            <element>
                <name>user</name>
                <value>wlevs</value>
            </element>
            <element>
                <name>password</name>
                <value>wlevs</value>
            </element>
        </properties>
    </driver-params>
    <connection-pool-params>
        <initial-capacity>5</initial-capacity>
        <max-capacity>10</max-capacity>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <test-frequency-seconds>5</test-frequency-seconds>
    </connection-pool-params>
    <data-source-params>
        <jndi-names>
            <element>orads</element>
        </jndi-names>
        <global-transactions-protocol>None</global-transactions-protocol>
    </data-source-params>
</data-source>
```

7.13 driver-params

Use the `driver-params` element to specify JDBC driver-related data-source parameters. This element has the following child elements and no attributes.

Table 7-8 Child Elements of: driver-params

XML Tag	Type	Description
`use-xa-data-source-interface`	`boolean`	Specifies that the server should use the XA interface of the JDBC driver. If the JDBC driver class used to create database connections implements both XA and non-XA versions of a JDBC driver, you can set this attribute to indicate that the server should treat the JDBC driver as an XA driver or as a non-XA driver. Default: `true`.
`password`	`string`	The password attribute passed to the JDBC driver when creating physical database connections.
`driver-name`	`string`	The full package name of JDBC driver class used to create the physical database connections in the connection pool in the data source.
`url`	`string`	The URL of the database to connect to. The format of the URL varies by JDBC driver. The URL is passed to the JDBC driver to create the physical database connections.
`properties`	`string`	Specifies the list of properties passed to the JDBC driver when creating physical database connections. This element contains one or more `element` child elements that contain child elements: • `name`: the property name. • `value`: the property value.

The following example shows how to use the `driver-params` element in the Oracle Event Processing server configuration file:

```
<data-source>
    <name>orads</name>
    <xa-params>
        <keep-xa-conn-till-tx-complete>true</keep-xa-conn-till-tx-complete>
    </xa-params>
    <driver-params>
        <url>jdbc:oracle:thin:@localhost:1521:ce102</url>
        <driver-name>oracle.jdbc.OracleDriver</driver-name>
        <properties>
            <element>
                <name>user</name>
                <value>wlevs</value>
            </element>
            <element>
                <name>password</name>
                <value>wlevs</value>
            </element>
        </properties>
    </driver-params>
    <connection-pool-params>
        <initial-capacity>5</initial-capacity>
        <max-capacity>10</max-capacity>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <test-frequency-seconds>5</test-frequency-seconds>
    </connection-pool-params>
    <data-source-params>
        <jndi-names>
            <element>orads</element>
        </jndi-names>
        <global-transactions-protocol>None</global-transactions-protocol>
    </data-source-params>
</data-source>
```

7.14 domain

Use the `domain` element to configure a domain name in the Oracle Event Processing server. This element has the name child element and no attributes.

The following example shows how to use the `domain` element in the Oracle Event Processing server configuration file. In the example, the domain's unique identifier is `WLEventServerDomain`.

```
<domain>
    <name>WLEventServerDomain</name>
</domain>
```

7.15 debug

Use the `debug` element to configure one or more debug properties for the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-9 Child Elements of: debug

XML Tag	Type	Description
`name`	`string`	The name of this `debug` configuration. For more information, see name.

Table 7-9 (Cont.) Child Elements of: debug

XML Tag	Type	Description
`debug-properties`	`string`	One or more child elements formed by taking a debug flag name (without its package name) and specifying a value of `true`.

The following example shows how to use the `debug` element to turn on Simple Declarative Services (SDS) debugging using debug flag `com.bea.core.debug.DebugSDS` in the Oracle Event Processing server configuration file.

```
<debug>
    <name>myDebug</name>
    <debug-properties>
        <DebugSDS>true</DebugSDS>
...
    </debug-properties>
</debug>
```

7.16 event-inspector

Use the `event-inspector` element to test a component. This element has the `pubsub-server-name` child element and no attributes.

The `pubsub-server-name` value is the value of the `http-pubsub` element name child element as defined in the local Oracle Event Processing server file.

The following example shows how to use the `event-inspector` element in a configuration file.

```
<event-inspector>
    <name>myEventInspectorConfig</name>
    <pubsub-server-name>myPubSub</pubsub-server-name>
</event-inspector>
```

The following example shows the corresponding local Oracle Event Processing server file entry:

```
<http-pubsub>
  <name>myPubSub</name>
  <path>/pubsub</path>
  <pub-sub-bean>
    <server-config>
      <supported-transport>
        <types>
          <element>long-polling</element>
        </types>
      </supported-transport>
      <publish-without-connect-allowed>true</publish-without-connect-allowed>
    </server-config>
    <channels>
        ...
    </channels>
  </pub-sub-bean>
</http-pubsub>
```

7.17 event-store

Use the event-store element to configure an event store for the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-10 Child Elements of: event-store

XML Tag	Type	Description
`name`	`string`	The name of this `debug` configuration. For more information, see name.
`provider-order`	`string`	Specifies the name of one or more `provider` child elements in the order in which the Oracle Event Processing server should access them. For more information, see: • rdbms-event-store-provider

The following example shows how to use the `event-store` element in the Oracle Event Processing server configuration file. In the example, the adapter's unique identifier is `myEventStore`.

```
<config>
    <event-store>
        <name>myEventStore</name>
        <provider-order>
            <provider>provider1</provider>
            <provider>provider2</provider>
        </provider-order>
    </event-store>
</config>
```

7.18 exported-jndi-context

Use the `exported-jndi-context element` to export a remote JNDI service that may be accessed via clients using RMI. It registers the JNDI context with the RMI service, so that it may be accessed remotely by clients that pass a provider URL parameter when they create their `InitialContext` object. This service requires that a jndi-context configuration object also be specified. If it is not, then this service will not be able to start.

This element has the following child elements and no attributes.

Table 7-11 Child Elements of: exported-jndi-context

XML Tag	Type	Description
`name`	`string`	The name of this `debug` configuration. For more information, see name.
`rmi-service-name`	`string`	The name of the RMI service that should be used to serve this JNDI context over the network. It must match an existing RMI object in the configuration. For more information, see rmi.

The following example shows how to use the `exported-jndi-context` element in the Oracle Event Processing server configuration file. In the example, the adapter's unique identifier is `RemoteJNDI`.

```
<rmi>
    <name>myRMI</name>
    <http-service-name>TestJetty</http-service-name>
</rmi>

<exported-jndi-context>
    <name>RemoteJNDI</name>
    <rmi-service-name>myRMI</rmi-service-name>
</exported-jndi-context>
```

7.19 glassfish-ws

Use the `glassfish-ws` element to configure web services in Oracle Event Processing. This element has the `http_service_name` element and no attributes.

Use the `http_service_name` element to specify the name of the HTTP service to use to register web service end points in Oracle Event Processing. The service is provided by a Jetty instance of the same name.

The following example shows how to use the `glassfish-ws` element in the Oracle Event Processing server configuration file. In the example, the `glassfish-ws` element's unique identifier is `myWS`.

```
<glassfish-ws>
    <name>myWS</name>
    <http-service-name>TestJetty</http-service-name>
</glassfish-ws>
```

7.20 http-pubsub

Use the `http-pubsub` element to configure an HTTP publish-subscribe service. This element has the following child elements and no attributes.

- name
- path
- pubsub-bean

The following example shows how to use the `http-pubsub` element in the Oracle Event Processing server configuration file. In the example, the `http-pubsub` element's unique identifier is `myPubsub`.

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
        <server-config>
            <supported-transport>
                <types>
                    <element>long-polling</element>
                </types>
            </supported-transport>
            <publish-without-connect-allowed>
                true
            </publish-without-connect-allowed>
        </server-config>
        <channels>
            <element>
                <channel-pattern>/evsmonitor</channel-pattern>
            </element>
```

```
            <element>
                <channel-pattern>/evsalert</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsdomainchange</channel-pattern>
            </element>
        </channels>
    </pub-sub-bean>
</http-pubsub>
```

7.21 jetty

Use the `jetty` element to configure an instance of the Jetty HTTP server. This element has the following child elements and no attributes.

Table 7-12 Child Elements of: jetty

XML Tag	Type	Description
`name`	`string`	The name of this `jetty` element. For more information, see name.
`network-io-name`	`string`	The name of the Network I/O service that should be used. This also defines which port the server listens on. This parameter must refer to the name of a valid "netio" configuration object.
`work-manager-name`	`string`	The name of the Work Manager that should be used for thread pooling. If this parameter is not specified, then Jetty will use a default work manager. For more information, see work-manager.
`scratch-directory`	`string`	The name of a directory where temporary files required for Web applications, JSPs, and other types of Web artifacts are kept. This parameter is overridden by the `scratch-directory` parameter on the jetty-web-app element. If this directory does not exist, it will be created.
`debug-enabled`	`boolean`	Enable debugging in the Jetty code. Specified debug messages are logged the same way as all other `Debug` level messages in the log service.
`listen-port`	`int`	The name of the network port that should be set. This parameter may not be set if the `network-io-name` parameter is not specified. When this parameter is used instead of `network-io-name`, a simplified socket I/O subsystem is used that does not require the `netio` module.
`secure-network-io-name`	`string`	The name of the Network I/O service that should be used for secure communications. The specified service must be configured to support SSL encryption. This parameter must refer to the name of a valid `netio` configuration object.

The following example shows how to use the `jetty` element in the Oracle Event Processing server configuration file. In the example, the `jetty` element's unique identifier is `TestJetty`.

```
<jetty>
    <name>TestJetty</name>
    <work-manager-name>WM</work-manager-name>
    <network-io-name>Netio</network-io-name>
    <secure-network-io-name>SecureNetio</secure-network-io-name>
    <debug-enabled>false</debug-enabled>
    <scratch-directory>JettyWork</scratch-directory>
</jetty>
```

7.22 jetty-web-app

Use the `jetty-web-app` element to represent a Web application for use by Jetty. Each instance of this object represents a Web application which must be deployed using the Jetty service.

This element has the following child elements and no attributes.

Table 7-13 Child Elements of: jetty-web-app

XML Tag	Type	Description
`name`	`string`	The name of this `jetty-web-app` element. For more information, see name.
`context-path`	`string`	The context path where this web app will be deployed in the web server's name space. Default:`/`
`scratch-directory`	`string`	The location where Jetty should store temporary files for this web app. This parameter overrides the `scratch-directory` parameter on the jetty element. If this directory does not exist, it will be created.
`path`	`string`	A file name that points to the location of the web app on the server. It may be a directory or a WAR file.
`jetty-name`	`string`	The name of the Jetty service where this Web application should be deployed. This name must match the name of an existing `jetty` configuration object. For more information, see jetty.

The following example shows how to use the `jetty-web-app` element in the Oracle Event Processing server configuration file. In the example, the `jetty-web-app` element's unique identifier is `financial`.

```
<jetty-web-app>
    <name>financial</name>
    <context-path>/financial</context-path>
    <path>../testws2/financialWS.war</path>
    <jetty-name>TestJetty</jetty-name>
</jetty-web-app>
```

7.23 jmx

Use the `jmx` element to configure Java Management Extension (JMX) properties in the Oracle Event Processing server.

Table 7-14 Child Elements of: jmx

XML Tag	Type	Description
`name`	`string`	The name of this `debug` configuration. For more information, see name.
`rmi-service-name`	`string`	The name of the RMI service that should be used to serve this JNDI context over the network. It must match an existing RMI object in the configuration. For more information, see rmi.
`jndi-service-name`	`string`	The name of the JNDI service to which the JMX server will bind its object.

The following example shows how to use the `jmx` element in the Oracle Event Processing server configuration file. In the example, the `jmx` element's unique identifier is `myJMX`.

```
<jmx>
    <name>myJMX</name>
    <jndi-service-name>JNDI</jndi-service-name>
    <rmi-service-name>RMI</rmi-service-name>
</jmx>
```

7.24 jndi-context

Use the `jndi-context` element to configure the JNDI provider. When it is placed in the configuration, the MSA JNDI Context is initialized. One instance of this configuration type must be placed in the configuration if the JNDI service is to be used, either locally, or remotely through the `exported-jndi-context` configuration type.

This element has the following child elements and no attributes.

Table 7-15 Child Elements of: jndi-context

XML Tag	Type	Description
`name`	`string`	The name of this `debug` configuration. For more information, see name.
`default-provider`	`string`	This parameter defaults to `true`. If it is set to `false` then the provider will not be installed as the default. The provider will be set as the default as long as there is at least one `JNDIContextType` bean in the configuration with `DefaultProvider` set to `true`. If multiple `JNDIContextType` objects are placed in the configuration, the effect will be the same as if one was started.

The following example shows how to use the `jndi-context` element in the Oracle Event Processing server configuration file. In the example, the adapter's unique identifier is `myJNDI`.

```
<jndi-context>
    <name>myJNDI</name>
    <default-provider>true</default-provider>
</jndi-context>
```

7.25 log-file

Use the `log-file` element to configure logging to a file on the Oracle Event Processing server.

Table 7-16 Child Elements of: log-file

XML Tag	Type	Description
`name`	`string`	The name of this `work-manager` element. For more information, see name.
`number-of-files-limited`	`boolean`	Determines whether old rotated files need to be kept around forever. Default: `false`.
`rotation-type`	`string`	Determines how the log file rotation will be performed based on size, time or not at all. Valid values: • `bySize` • `byTime` • `none` Default: `bySize`.
`rotation-time`	`string`	The time in `k:mm` format when the first rotation happens where `k` is the hour specified in 24-hour notation and `mm` is the minutes. Default: 00:00.
`rotated-file-count`	`int`	If old rotated files are to be deleted, this parameter determines how many of the last files to always keep. Default: 7.
`rotation-size`	`int`	The size threshold at which the log file is rotated in KB. Default: 500.
`rotation-time-span-factor`	`long`	The factor that is applied to the timespan to arrive at the number of milliseconds that will be the frequency of time based log rotations. Default: (`long`)(3600 * 1000).
`rotation-time-span`	`int`	The interval for every time based log rotation. Default: 24.
`base-log-file-name`	`string`	Log file name. Default: `server.log`
`rotate-log-on-startup-enabled`	`boolean`	Specifies whether the log file will be rotated on startup. Default: `true`.

Table 7-16 (Cont.) Child Elements of: log-file

XML Tag	Type	Description
`log-file-severity`	`string`	Specifies the threshold importance of the messages that are propagated to the handlers. The default is `Info` so that to see `Debug` and `Trace` messages you need to ensure that the severity is set to either `Debug` or `Trace`. Valid values: • `Emergency` • `Alert` • `Critical` • `Error` • `Warning` • `Notice` • `Info` • `Debug` • `Trace` Default: `Notice`.
`log-file-rotation-dir`	`string`	The directory where the old rotated files are stored. If not set, the old files are stored in the same directory as the base log file.

The following example shows how to use the `log-file` element in the Oracle Event Processing server configuration file. In the example, the `log-file` element's unique identifier is `logFile`.

```
<log-file>
    <name>logFile</name>
    <number-of-files-limited>true</number-of-files-limited>
    <rotated-file-count>4</rotated-file-count>
    <rotate-log-on-startup-enabled>true</rotate-log-on-startup-enabled>
</log-file>
```

7.26 log-stdout

Use the `log-stdout` element to configure logging to standard out (console) on the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-17 Child Elements of: log-stdout

XML Tag	Type	Description
`name`	`string`	The name of this `work-manager` element. For more information, see name.
`stack-trace-depth`	`int`	Determines the number of stack trace frames to display on standard out. All frames are displayed in the log file. A value of -1 means all frames are displayed. Default: -1.
`stack-trace-enabled`	`boolean`	Specifies whether to dump stack traces to the console when included in a logged message. Default: `true`.

Table 7-17 (Cont.) Child Elements of: log-stdout

XML Tag	Type	Description
`stdout-severity`	string	Defines the threshold importance of the messages that are propagated to the handlers. The default is `Info` so that to see `Debug` and `Trace` messages you need to ensure that the severity is set to either `Debug` or `Trace`. Valid values: • `Emergency` • `Alert` • `Critical` • `Error` • `Warning` • `Notice` • `Info` • `Debug` • `Trace` Default: `Notice`.

The following example shows how to use the `log-stdout` element in the Oracle Event Processing server configuration file. In the example, the `log-stdout` element's unique identifier is `logStdout`.

```
<log-stdout>
    <name>logStdout</name>
    <stdout-severity>Debug</stdout-severity>
</log-stdout>
```

7.27 logging-service

Use the logging-service element to configure a logging service on the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-18 Child Elements of: logging-service

XML Tag	Type	Description
`name`	`string`	The name of this `work-manager` element. For more information, see name.
`log-file-config`	`string`	The configuration of the log file and its rotation policies.
`stdout-config`	`string`	The configuration of the stdout output.

Table 7-18 (Cont.) Child Elements of: logging-service

XML Tag	Type	Description
`logger-severity`	`string`	Defines the threshold importance of the messages that are propagated to the handlers. The default is `Info` so that to see `Debug` and `Trace` messages you need to ensure that the severity is set to either `Debug` or `Trace`. Valid values: • `Emergency` • `Alert` • `Critical` • `Error` • `Warning` • `Notice` • `Info` • `Debug` • `Trace` Default: `Info`.
`logger-severity-properties`	See Description	The Severity values for different stages in the Logger tree composed of one or more `entry` child elements each containing a `key` and `value` child element.

The following example shows how to use the `logging-service` element in the Oracle Event Processing server configuration file. In the example, the `logging-service` element's unique identifier is `myLogService`.

```
<logging-service>
    <name>myLogService</name>
    <stdout-config>myStdoutConfig</stdout-config>
    <logger-severity>Notice</logger-severity>
    <logger-severity-properties>
      <entry>
        <key>FileAdapter</key>
        <value>Debug</value>
      </entry>
      <entry>
        <key>CQLProcessor</key>
        <value>Debug</value>
      </entry>
    </logger-severity-properties>
</logging-service>
```

7.28 message-filters

Use the `message-filters` element to configure one or more message filters for a pubsub-bean element.

This element has one or more `element` child elements that each contain a `message-filter-name` and `message-filter-class` child element.

This element has no attributes.

The following example shows how to use the `message-filters` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>pubsub</name>
```

```
    <path>/pubsub</path>
    <pub-sub-bean>
...
      <message-fitlers>
        <element>
          <message-filter-name>Foo</message-filter-name>
          <message-filter-class>Foo</message-filter-class>
        </element>
        <element>
          <message-filter-name>Foo</message-filter-name>
          <message-filter-class>Foo</message-filter-class>
        </element>
      </message-filters>
...
    </pub-sub-bean>
</http-pubsub>
```

7.29 name

Use the `name` element to declare a unique identifier for an Oracle Event Processing server configuration element. This element has no child elements and no attributes.

The following example shows how to use the `name` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>pubsub</name>
    <path>/pubsub</path>
...
</http-pubsub>
```

7.30 netio

Use the `netio` element to represent a network input and output (IO) service, that can be used by other services to act as the server for network IO. This element has the following child elements and no attributes.

Table 7-19 Child Elements of: netio

XML Tag	Type	Description
`name`	`string`	The name of this `netio` element. For more information, see name.
`ssl-config-bean-name`	`string`	The name of the SSL configuration object to use. If not null, then this client will create secure sockets using the specified SSL configuration. If not set, then no SSL will be supported.
`provider-type`	`string`	Specify which provider to use for the underlying socket implementation.
`io-threads`	`int`	A hint to the provider as to the number of threads to use for processing sockets. A value of zero will result in the provider choosing based on its own default.Default: 0.
`port`	`int`	The port to listen on. The server will immediately start to listen for incoming connections on this port.

Table 7-19 (Cont.) Child Elements of: netio

XML Tag	Type	Description
`listen-address`	`string`	The address on which this instance of Netio should listen for incoming connections. It may be set to a numeric IP address in the `a.b.c.d` format, or to a host name. If not set, then `netio` will listen on all network interfaces. Note that the value of this parameter cannot be validated until the server actually starts.

The following example shows how to use the `netio` element in the Oracle Event Processing server configuration file. In the example, the `netio` element's unique identifier is `myNetio`.

```
<netio>
    <name>myNetio</name>
    <port>12345</port>
</netio>
```

7.31 netio-client

Use the `netio-client` element to register a network input/output (IO) service that may be used to perform non-blocking network IO, but which will not act as a server and listen for incoming connections. This element has the following child elements and no attributes.

Table 7-20 Child Elements of: netio-client

XML Tag	Type	Description
`name`	`string`	The name of this `netio` element. For more information, see name.
`ssl-config-bean-name`	`string`	The name of the SSL configuration object to use. If not null, then this client will create secure sockets using the specified SSL configuration. If not set, then no SSL will be supported.
`provider-type`	`string`	Specify which provider to use for the underlying socket implementation.

The following example shows how to use the `netio-client` element in the Oracle Event Processing server configuration file. In the example, the `netio-client` element's unique identifier is `netiossl`.

```
<netio-client>
    <name>netiossl</name>
    <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
    <provider-type>NIO</provider-type>
</netio-client>
```

7.32 partition-order-capacity

Use the `partition-order-capacity` element to define the maximum capacity of a query partition when the ordering-constraint attribute is set to PARTITION_ORDERED. Set this element on a cql component. Consider setting this

element's value when you've configured a query processor for parallel execution, and when the query's ordering-constraint attribute is set to PARTITION_ORDERED. The default value is 4.

This element has no child elements and no attributes.

The following example shows how to use the `partition-order-capacity` element in the Oracle Event Processing server configuration file:

```
<cql>
    <name>myCQL</name>
    <partition-order-capacity>20</partition-order-capacity>
</cql>
```

7.33 path

Use the `path` element to configure the path for an http-pubsub element. This element has no child elements or attributes.

The following example shows how to use the `path` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
        <server-config>
            <supported-transport>
                <types>
                    <element>long-polling</element>
                </types>
            </supported-transport>
            <publish-without-connect-allowed>
                true
            </publish-without-connect-allowed>
        </server-config>
        <channels>
            <element>
                <channel-pattern>/evsmonitor</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsalert</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsdomainchange</channel-pattern>
            </element>
        </channels>
    </pub-sub-bean>
</http-pubsub>
```

7.34 pubsub-bean

Use the `pubsub-bean` element to configure a publish-subscribe bean for an http-pubsub element. This element has the following child elements and no attributes.

- name
- server-config
- message-filters

- channels
- channel-constraints
- services

See http://www.oracle.com/webfolder/technetwork/weblogic/weblogic-pubsub/1.0/weblogic-pubsub.xsd.

The following example shows how to use the `pubsub-bean` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>myPubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
        <server-config>
            <supported-transport>
                <types>
                    <element>long-polling</element>
                </types>
            </supported-transport>
            <publish-without-connect-allowed>
                true
            </publish-without-connect-allowed>
        </server-config>
        <channels>
            <element>
                <channel-pattern>/evsmonitor</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsalert</channel-pattern>
            </element>
            <element>
                <channel-pattern>/evsdomainchange</channel-pattern>
            </element>
        </channels>
    </pub-sub-bean>
</http-pubsub>
```

7.35 rdbms-event-store-provider

Use the `rdbms-event-store-provider` element to configure an event store provider that uses a relational database management system in the Oracle Event Processing server. By default, Oracle Event Processing server uses a Berkeley database instance as the event store provider as bdb-config describes.

This element has the following child elements and no attributes.

Table 7-21 Child Elements of: rdbms-event-store-provider

XML Tag	Type	Description
`name`	`string`	The name of this `debug` configuration. For more information, see name.
`init-timeout`	`int`	The maximum time (in milliseconds) that the Oracle Event Processing server will wait for this provider to initialize. Default: 10000 ms.

Table 7-21 (Cont.) Child Elements of: rdbms-event-store-provider

XML Tag	Type	Description
`data-source-name`	`string`	The name of a data source element. For more information, see data-source.
`user-policy-attributes`	See Description	One or more `entry` child elements that each contain a `key` and `value` child element that you use to specify additional data source properties.

The following example shows how to use the `rdbms-event-store-provider` element in the Oracle Event Processing server configuration file:

```
<rdbms-event-store-provider>
    <name>test-rdbms-provider</name>
    <init-timeout>10000</init-timeout>
    <data-source-name>derby1</data-source-name>
    <user-policy-attributes>
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
            <key>key1</key>
            <value>value1</value>
        <entry>
        </entry>
    </user-policy-attributes>
</rdbms-event-store-provider>
```

7.36 rmi

Use the `rmi` element to configure an RMI service, which allows server- side objects to be exported to remote clients. This element has the following child elements and no attributes.

Table 7-22 Child Elements of: rmi

XML Tag	Type	Description
`name`	`string`	The name of this `rmi` element. For more information, see name.
`heartbeat-period`	`int`	The number of failed heartbeat attempts before triggering disconnect notifications to all registered listeners.Default-Value: 4.
`http-service-name`	`string`	The name of the HTTP service that this service should use to register remote objects. The service may be provided by a Jetty or Tomcat instance of the same name.
`heartbeat-interval`	`int`	The time in milliseconds between heartbeats. Once the number of unsuccessful heartbeat attempts has reached the value specified by the `HeartbeatPeriod` attribute, all registered `DisconnectListener` instances will be notified. Default-Value: 5000.

The following example shows how to use the `rmi` element in the Oracle Event Processing server configuration file. In the example, the `rmi` element's unique identifier is `myRMI`.

```
<rmi>
    <name>myRMI</name>
    <http-service-name>TestJetty</http-service-name>
</rmi>
```

7.37 scheduler

Use the `scheduler` element to configure cql scheduler options in the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-23 Child Elements of: scheduler

XML Tag	Type	Description
`time-slice`	`int`	The frequency at which the Oracle CQL scheduler executes Oracle CQL queries. Default: 1000 ms
`schedule-on-new-thread`	`boolean`	Whether or not the Oracle Event Processing Service Engine scheduler will use a separate thread. Options are: • `true`: the scheduler runs in a separate thread. • `false`: the scheduler runs in the same thread as the Oracle Event Processing Service Engine (Default).

The following example shows how to use the `scheduler` element in the Oracle Event Processing server configuration file:

```
<cql>
    <name>myCQL</name>
    <calendar>
        <date-format>myclass</date-format>
        <timezone>10</timezone>
    </calendar>
    <scheduler>
        <class-name>oracle.cep.execution.scheduler.FIFOScheduler</class-name>
        <threads>10</threads>
        <direct-interop>false</direct-interop>
    </scheduler>
</cql>
```

7.38 server-config

Use the `server-config` element to configure the server-specific properties of a pubsub-bean element. This element has the following child elements and no attributes.

Table 7-24 Child Elements of: server-config

XML Tag	Type	Description
`name`	`string`	The name of this `server-config` element. For more information, see name.

Table 7-24 (Cont.) Child Elements of: server-config

XML Tag	Type	Description
`supported-transport`	See Description	This element contains one or more `types` child elements, one for each supported transport. Each `types` child element contains an `element` child element with the transport name as a `string` value. Valid values: • `long-polling`: Using this transport, the client requests information from Oracle Event Processing server and if Oracle Event Processing server does not have information available, it does not reply until it has. When the Oracle Event Processing server replies, the client typically sends another request immediately. • `callback-polling`: Use this transport for HTTP publish-subscribe applications using a cross domain configuration in which the browser downloads the page from one Web server (including the JavaScript code) and connects to another server as an HTTP publish-subscribe client. This is required by the Bayeux protocol. For more information on the Bayeux protocol, see `http://svn.cometd.org/trunk/bayeux/bayeux.html`.
`client-timeout-secs`	`int`	Specifies the number of seconds after which the HTTP pub-sub server disconnects a client if the client does has not sent back a connect/reconnect message. Default: 60.
`persistent-client-timeout-secs`	`int`	Specifies the number of seconds after which persistent clients are disconnected and deleted by the pub-sub server, if during that time the persistent client does not send a connect or re-connect message. This value must be larger than client-timeout-secs. If the persistent client reconnects before the persistent timeout is reached, the client receives all messages that have been published to the persistent channel during that time; if the client reconnects after the timeout, then it does not get the messages. Default: 600 seconds.
`interval-millisecs`	`int`	Specifies how long (in milliseconds) the client can delay subsequent requests to the `/meta/connect` channel. Default: 500 ms.
`work-manager`	`string`	Specifies the name of the work manager that delivers messages to clients. The value of this element corresponds to the value of the `name` child element of the `work-manager` you want to assign. For more information, see work-manager.

Table 7-24 (Cont.) Child Elements of: server-config

XML Tag	Type	Description
`publish-without-connect-allowed`	`boolean`	Specifies whether clients can publish messages without having explicitly connected to the HTTP pub-sub server. Valid values: • `true` • `false`

The following example shows how to use the `server-config` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>pubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
      <server-config>
        <name>/pubsub</name>
        <supported-transport>
          <types>
            <element>long-polling</element>
          </types>
        </supported-transport>
        <publish-without-connect-allowed>true</publish-without-connect-allowed>
      </server-config>
    <channels>
        <element>
          <channel-pattern>/evsmonitor</channel-pattern>
        </element>
        <element>
          <channel-pattern>/evsalert</channel-pattern>
        </element>
        <element>
          <channel-pattern>/evsdomainchange</channel-pattern>
        </element>
      </channels>
    </pub-sub-bean>
</http-pubsub>
```

7.39 services

Use the `services` element to configure the service properties of a pubsub-bean element. This element has the following child elements and no attributes.

Table 7-25 Child Elements of: services

XML Tag	Type	Description
`service-channel`	`string`	Specifies a service channel, for example: `/service/echo`.
`service-class`	`string`	Specifies the class to service this service, for example: `EchoService`.

Table 7-25 (Cont.) Child Elements of: services

XML Tag	Type	Description
`service-method`	`string`	Define a service method in the service class. The service method must have only one payload parameter of type `Object`. For example: `Object echo(Object payload)`.

The following example shows how to use the `services` element in the Oracle Event Processing server configuration file:

```
<http-pubsub>
    <name>pubsub</name>
    <path>/pubsub</path>
    <pub-sub-bean>
      <server-config>
        <name>/pubsub</name>
        <supported-transport>
          <types>
            <element>long-polling</element>
          </types>
        </supported-transport>
        <publish-without-connect-allowed>true</publish-without-connect-allowed>
      </server-config>
    <channels>
        <element>
          <channel-pattern>/evsmonitor</channel-pattern>
        </element>
        <element>
          <channel-pattern>/evsalert</channel-pattern>
        </element>
        <element>
          <channel-pattern>/evsdomainchange</channel-pattern>
        </element>
      </channels>
      <services>
          <element>
              <service-channel>Foo</service-channel>
              <service-class>Foo</service-class>
              <service-method>Foo</service-method>
          </element>
      </services>
    </pub-sub-bean>
</http-pubsub>
```

7.40 show-detail-error-message

Use the `show-detail-error-message` element to configure whether or not the Oracle Event Processing server uses secure connections. This element has the following child elements and no attributes.

Table 7-26 Child Elements of: show-detail-error-message

XML Tag	Type	Description
`name`	`string`	The name of this `show-detail-error-message` element. For more information, see name.
`value`	`boolean`	Whether or not to show detailed error messages. Valid values: • `true`: the Oracle Event Processing server shows detailed error messages. • `false`: the Oracle Event Processing server shows abbreviated error messages (default).

The following example shows how to use the `show-detail-error-message` element in the Oracle Event Processing server configuration file. In the example, the `show-detail-error-message` element's unique identifier is `myShowDetail`.

```
<show-detail-error-message>
    <name>myShowDetail</name>
    <value>true</value>
</show-detail-error-message>
```

7.41 ssl

Use the `ssl` element to configure Secure Sockets Layer-specific properties on the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-27 Child Elements of: ssl

XML Tag	Type	Description
`name`	`string`	The name of this cluster. For more information, see name.
`key-store`	`string`	Specifies the file path to the key store such as `./ssl/evsidentity.jks`.
`key-store-pass`	See Description	This element contains a `password` child element with a `string` value that specifies the password used to access the key store.
`key-store-alias`	`string`	Specifies the alias for the key store.
`key-manager-algorithm`	`string`	Specifies the key manager algorithm such as `SunX509`.
`ssl-protocol`	`string`	Specifies the SSL protocol such as `TLS`.
`trust-store`	`string`	Specifies the file path to the trust store such as `./ssl/evstrust.jks`.
`trust-store-pass`	See Description	This element contains a `password` child element with a `string` value that specifies the password used to access the trust store.

Table 7-27 (Cont.) Child Elements of: ssl

XML Tag	Type	Description
`trust-store-alias`	`string`	Specifies the alias for the trust store.
`trust-store-type`	`string`	Specifies the trust store type such as `JKS`.
`trust-manager-algorithm`	`string`	Specifies the trust manager algorithm such as `SunX509`.
`enforce-fips`	`boolean`	Specifies whether or not Oracle Event Processing server uses a Federal Information Processing Standards (FIPS)-certified pseudo-random number generator.
`need-client-auth`	`boolean`	Specifies whether or not client certificate authentication is required.
`ciphers`	See Description	This element contains one or more `cipher` child elements, each with a `string` value that specifies the ciphers that are required.
`secure-random-algorithm`	`string`	When `enforce-fips` is set to `true`, specify the secure random algorithm to use. Valid values: • `FIPS186PRNG`
`secure-random-provider`	`string`	When `enforce-fips` is set to `true`, specify the secure random provider to use. Valid values: • `JsafeJCE`

The following example shows how to use the `ssl` element in the Oracle Event Processing server configuration file:

In the example, the `ssl` element's unique identifier is `sslConfig`.

```
<ssl>
    <name>sslConfig</name>
    <key-store>./ssl/evsidentity.jks</key-store>
    <key-store-pass>
        <password>{Salted-3DES}s4YUEvH4Wl2DAjb45iJnrw==</password>
    </key-store-pass>
    <key-store-alias>evsidentity</key-store-alias>
    <key-manager-algorithm>SunX509</key-manager-algorithm>
    <ssl-protocol>TLS</ssl-protocol>
    <enforce-fips>false</enforce-fips>
    <need-client-auth>false</need-client-auth>
</ssl>
```

7.42 timeout-seconds

Use the `timeout-seconds` element to configure weblogic-jta-gateway default transaction time out in seconds in the Oracle Event Processing server. The default: 60. this element has no child elements and no attributes.

The following example shows how to use the `timeout-seconds` element in the Oracle Event Processing server configuration file:

```
<weblogic-jta-gateway>
    <name>myJTAGateway</name>
    <timeout-seconds>90</timeout-seconds>
    <weblogic-instances>
        <weblogic-instance>
            <domain-name>ocep_domain</domain-name>
            <server-name>fxserver</server-name>
            <protocol>t3</protocol>
            <host-address>ariel</host-address>
            <port>9002</port>
        </weblogic-instance>
    </weblogic-instances>
</weblogic-jta-gateway>
```

7.43 transaction-manager

Use the `transaction-manager` element to configure transaction manager properties in the Oracle Event Processing server. This element has the following child elements and no attributes.

Table 7-28 Child Elements of: transaction-manager

XML Tag	Type	Description
`name`	`string`	The name of this `transaction-manager` element. For more information, see name.
`max-resource-requests-on-server`	`int`	Maximum number of concurrent requests to resources allowed for each server. Default: 50.
`max-resource-unavailable-millis`	`long`	Maximum duration in milliseconds that a resource is declared dead. After the duration, the resource will be declared available again, even if the resource provider does not explicitly re-register the resource.Default: 1800000.
`security-interop-mode`	`string`	Specifies the security mode of the communication channel used for XA calls between servers that participate in a global transaction. All server instances in a domain must have the same security mode setting. Valid values: • `default`: The transaction coordinator makes calls using the kernel identity over an admin channel if it is enabled, and anonymous otherwise. Man-in-the-middle attacks are possible if the admin channel is not enabled. • `Performance`: The transaction coordinator makes calls using anonymous at all times. This implies a security risk since a malicious third party could then try to affect the outcome of transactions using a man-in-the-middle attack. • `Compatibility`: The transaction coordinator makes calls as the kernel identity over an insecure channel. This is a high security risk because a successful man-in-the-middle attack would allow the attacker to gain administrative control over both domains. This setting should only be used when strong network security is in place. Default: `default`.

Table 7-28 (Cont.) Child Elements of: transaction-manager

XML Tag	Type	Description
`parallel-xa-enabled`	`boolean`	Execute XA calls in parallel if there are available threads. Default: `true`.
`tlog-location`	`string`	The location of the file store that contains the transaction log. This attribute can be either an absolute or relative path in the filesystem.
`max-xa-call-millis`	`long`	Maximum allowed duration of XA calls to resources. If a particular XA call to a resource exceeds the limit, the resource is declared unavailable. Default: 120000.
`timeout-seconds`	`int`	The default transaction timeout in seconds. Default: 30.
`checkpoint-interval-seconds`	`int`	The interval at which the transaction manager performs transaction log checkpoint operations. Default: 300.
`forget-heuristics`	`boolean`	Specifies whether the transaction manager will automatically perform an XAResource forget operation for heuristic transaction completions. When enabled, the transaction manager automatically performs an XA Resource `forget` operation for all resources as soon as the transaction learns of a heuristic outcome. Disable this feature only if you know what to do with the resource when it reports a heuristic decision. Default: `true`.
`before-completion-iteration-limit`	`int`	The maximum number of cycles that the transaction manager will perform the before completion synchronization callback processing. Default: 10.
`abandon-timeout-seconds`	`int`	The transaction abandon timeout seconds for transactions in the second phase of the two-phase commit (prepared and later). During the second phase of the two-phase commit process, the transaction manager will continue to try to complete the transaction until all resource managers indicate that the transaction is completed. Using this timeout, you can set the maximum time that a transaction manager will persist in attempting to complete a transaction during the second phase of the transaction. After the abandon transaction timer expires, no further attempt is made to resolve the transaction. If the transaction is in a prepared state before being abandoned, the transaction manager will roll back the transaction to release any locks held on behalf of the abandoned transaction. Default: 86400.
`serialize-enlistments-gc-interval-millis`	`long`	The interval at which internal objects used to serialize resource enlistment are cleaned up. Default: 30000.

Table 7-28 (Cont.) Child Elements of: transaction-manager

XML Tag	Type	Description
`unregister-resource-grace-period`	`int`	The grace period (number of seconds) that the transaction manager waits for transactions involving the resource to complete before unregistering a resource. The grace period can help minimize the risk of abandoned transactions because of an unregistered resource, such as a JDBC data source module packaged with an application. During the specified grace period, the unregisterResource call will block until the call can return, and no new transactions are started for the associated resource. If the number of outstanding transactions for the resource goes to 0, the unregisterResource call returns immediately. At the end of the grace period, if there are still outstanding transactions associated with the resource, the unregisterResource call returns and a log message is written on the server on which the resource was previously registered. Default: 30.
`rmi-service-name`	`string`	The name of the RMI service that is used for distributed transaction coordination. For more information, see rmi.
`max-unique-name-statistics`	`int`	The maximum number of unique transaction names for which statistics will be maintained. Default: 1000.
`purge-resource-from-checkpoint-interval-seconds`	`int`	The interval that a particular resource must be accessed within for it to be included in the checkpoint record. Default: 86400.
`max-transactions`	`int`	The maximum number of simultaneous in-progress transactions allowed on this server. Default: 10000.
`migration-checkpoint-interval-seconds`	`int`	The interval that the checkpoint is done for the migrated transaction logs (TLOGs). Default: 60.
`recovery-threshold-millis`	`long`	The interval that recovery is attempted until the resource becomes available. Default: 300000.
`max-transactions-health-interval-millis`	`long`	The interval for which the transaction map must be full for the JTA subsystem to declare its health as `CRITICAL`. Default: 60000.
`parallel-xa-dispatch-policy`	`string`	The dispatch policy to use when performing XA operations in parallel. By default the policy of the thread coordinating the transaction is used.

The following example shows how to use the `transaction-manager` element in the Oracle Event Processing server configuration file. In the example, the `transaction-manager` element's unique identifier is `My_tm`.

```
<transaction-manager>
    <name>My_tm</name>
    <timeout-seconds>30</timeout-seconds>
    <abandon-timeout-seconds>86400</abandon-timeout-seconds>
    <forget-heuristics>true</forget-heuristics>
    <before-completion-iteration-limit>12</before-completion-iteration-limit>
    <max-transactions>10100</max-transactions>
    <max-unique-name-statistics>500</max-unique-name-statistics>
    <max-resource-requests-on-server>50</max-resource-requests-on-server>
    <max-resource-unavailable-millis>1800000</max-resource-unavailable-millis>
    <recovery-threshold-millis>300000</recovery-threshold-millis>
    <max-transactions-health-interval-millis>
        60000
    </max-transactions-health-interval-millis>
    <purge-resource-from-checkpoint-interval-seconds>
        86400
    </purge-resource-from-checkpoint-interval-seconds>
    <checkpoint-interval-seconds>300</checkpoint-interval-seconds>
    <parallel-xa-enabled>true</parallel-xa-enabled>
    <unregister-resource-grace-period>30</unregister-resource-grace-period>
    <security-interop-mode>default</security-interop-mode>
    <rmi-service-name>RMI_ce1</rmi-service-name>
  </transaction-manager>
```

7.44 use-secure-connections

Use the `use-secure-conditions` element to configure whether or not the Oracle Event Processing server uses secure connections. This element has the following child elements and no attributes.

Table 7-29 Child Elements of: use-secure-connections

XML Tag	Type	Description
`name`	`string`	The name of this `use-secure-connections` element. For more information, see name.
`value`	`boolean`	Whether or not to use secure connections. Valid values: • `true`: the Oracle Event Processing server uses only secure connections. • `false`: the Oracle Event Processing server accepts connections that are not secure.

The following example shows how to use the `use-secure-connections` element in the Oracle Event Processing server configuration file:

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

7.45 user-event-store-provider

7.46 weblogic-instances

Use the `weblogic-instances` element to configure Oracle Event Processing server instances for a weblogic-jta-gateway element. This element has the following child elements and no attributes.

Table 7-30 Child Elements of: weblogic-instances

XML Tag	Type	Description
`domain-name`	`string`	Specifies the name of the domain of the Oracle Event Processing server.
`server-name`	`string`	Specifies the name of the Oracle Event Processing server.
`protocol`	`string`	Specifies the JTA protocol. Default: `t3`.
`host-address`	`string`	The host name or IP address of the Oracle Event Processing server.
`port`	`int`	The netio port for the Oracle Event Processing server.

The following example shows how to use the `weblogic-instances` element in the Oracle Event Processing server configuration file:

```
<weblogic-jta-gateway>
    <name>myJTAGateway</name>
    <timeout-seconds>90</timeout-seconds>
    <weblogic-instances>
        <weblogic-instance>
            <domain-name>ocep_domain</domain-name>
            <server-name>fxserver</server-name>
            <protocol>t3</protocol>
            <host-address>ariel</host-address>
            <port>9002</port>
        </weblogic-instance>
    </weblogic-instances>
</weblogic-jta-gateway>
```

7.47 weblogic-jta-gateway

Use the `weblogic-jta-gateway` element to configure the attributes for the singleton Oracle Event Processing server client JTA gateway service. This element has the following child elements and no attributes.

- name
- timeout-seconds
- weblogic-instances

The following example shows how to use the `weblogic-jta-gateway` element in the Oracle Event Processing server configuration file. In the example, the `weblogic-jta-gateway` element's unique identifier is `myJTAGateway`.

```
<weblogic-jta-gateway>
    <name>myJTAGateway</name>
```

```
    <timeout-seconds>90</timeout-seconds>
    <weblogic-instances>
        <weblogic-instance>
            <domain-name>ocep_domain</domain-name>
            <server-name>fxserver</server-name>
            <protocol>t3</protocol>
            <host-address>ariel</host-address>
            <port>9002</port>
        </weblogic-instance>
    </weblogic-instances>
</weblogic-jta-gateway>
```

7.48 weblogic-rmi-client

Use the `weblogic-rmi-client` element to configure the attributes for the singleton Oracle Event Processing server RMI client. This element has the following child elements and no attributes.

Table 7-31 Child Elements of: weblogic-rmi-client

XML Tag	Type	Description
`name`	`string`	The name of this `weblogic-rmi-client` element. For more information, see name.
`netio-name`	`string`	Specifies the name of the `netio-client` element to use. For more information, see netio-client.
`secure-netio-name`	`string`	Specifies the name of the `netio-client` element configured for SSL. For more information, see netio-client.

The following example shows how to use the `weblogic-rmi-client` element in the Oracle Event Processing server configuration file. In the example, the `weblogic-rmi-client` element's unique identifier is `wlclient`.

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

<netio-client>
    <name>netiossl</name>
    <provider-type>NIO</provider-type>
    <ssl-config-bean-name>sslConfig</ssl-config-bean-name>
</netio-client>

<weblogic-rmi-client>
    <name>wlclient</name>
    <netio-name>netio</netio-name>
    <secure-netio-name>netiossl</secure-netio-name>
</weblogic-rmi-client>
```

7.49 work-manager

Use the `work-manager` element to configure a work manager on the Oracle Event Processing server.

Table 7-32 Child Elements of: work-manager

XML Tag	Type	Description
`name`	`string`	The name of this `work-manager` element. For more information, see name.
`min-threads-constraint`	`int`	The minimum threads constraint this work manager should use. Default: -1.
`fairshare`	`int`	The fairshare value this work manager should use. Default: -1.
`max-threads-constraint`	`int`	The maximum threads constraint this work manager should use. Default: -1.

The following example shows how to use the `work-manager` element in the Oracle Event Processing server configuration file. In the example, the `work-manager` element's unique identifier is `WM`.

```
<work-manager>
    <name>WM</name>
    <fairshare>5</fairshare>
    <min-threads-constraint>1</min-threads-constraint>
    <max-threads-constraint>4</max-threads-constraint>
</work-manager>
```

7.50 xa-params

Use the `xa-params` element to specify distributed transaction-related data-source parameters. This element has the following child elements and no attributes.

Table 7-33 Child Elements of: xa-params

XML Tag	Type	Description
`keep-xa-conn-till-tx-complete`	`boolean`	Enables the server to associate the same XA database connection from the connection pool with a global transaction until the transaction completes. Only applies to connection pools that use an XA driver. Use this setting to work around specific problems with JDBC XA drivers. Default: `true`.

Table 7-33 (Cont.) Child Elements of: xa-params

XML Tag	Type	Description
`xa-transaction-timeout`	`int`	The number of seconds to set as the transaction branch timeout. If set, this value is passed as the transaction timeout value in the `XAResource.setTransactionTimeout` call on the XA resource manager, typically the JDBC driver. When this value is set to 0, the Transaction Manager passes the global server transaction timeout in seconds in the method. If set, this value should be greater than or equal to the global server transaction timeout. Note: You must enable `xa-set-transaction-timeout` to enable setting the transaction branch timeout. Default: 0.
`rollback-local-tx-upon-conn-close`	`boolean`	Enables the server to call `rollback` on the connection before returning the connection to the connection pool. Enabling this attribute will have a performance impact as the rollback call requires communication with the database server. Default: `false`.

Table 7-33 (Cont.) Child Elements of: xa-params

XML Tag	Type	Description
`xa-retry-duration-seconds`	`int`	Determines the duration in seconds for which the transaction manager will perform recover operations on the resource. A value of zero indicates that no retries will be performed. Default: 60.
`xa-set-transaction-timeout`	`boolean`	Enables the server to set a transaction branch timeout based on the value for `xa-transaction-timeout`. When enabled, the Transaction Manager calls `XAResource.setTransactionTimeout` before calling `XAResource.start`, and passes either the XA Transaction Timeout value or the global transaction timeout. You may want to set a transaction branch timeout if you have long-running transactions that exceed the default timeout value on the XA resource. Default: `false`.

Table 7-33 (Cont.) Child Elements of: xa-params

XML Tag	Type	Description
`keep-logical-conn-open-on-release`	`boolean`	Enables the server to keep the logical JDBC connection open for a global transaction when the physical XA connection is returned to the connection pool. Select this option if the XA driver used to create database connections or the DBMS requires that a logical JDBC connection be kept open while transaction processing continues (although the physical XA connection can be returned to the connection pool). Only applies to data sources that use an XA driver. Use this setting to work around specific problems with JDBC XA drivers. Default: `false`.
`resource-health-monitoring`	`boolean`	Enables JTA resource health monitoring for an XA data source. When enabled, if an XA resource fails to respond to an XA call within the period specified in MaxXACallMillis, the server marks the data source as unhealthy and blocks any further calls to the resource. This property applies to XA data sources only, and is ignored for data sources that use a non-XA driver. Default: `true`.

Table 7-33 (Cont.) Child Elements of: xa-params

XML Tag	Type	Description
`new-xa-conn-for-commit`	`boolean`	Specifies that a dedicated XA connection is used for commit and rollback processing for a global transaction. Only applies to data sources that use an XA driver. Use this setting to work around specific problems with JDBC XA drivers. Default: `false`.
`xa-end-only-once`	`boolean`	Specifies that `XAResource.end` is called only once for each pending `XAResource.start`. This option prevents the XA driver from calling `XAResource.end(TMSUSPEND)` and `XAResource.end(TMSUCCESS)` successively. Only applies to data sources that use an XA driver. Use this setting to work around specific problems with JDBC XA drivers. Default: `false`.
`xa-retry-interval-seconds`	`int`	The number of seconds between XA retry operations if XARetryDurationSeconds is set to a positive value. Default: 60.

Table 7-33 (Cont.) Child Elements of: xa-params

XML Tag	Type	Description
`recover-only-once`	`boolean`	Specifies that the transaction manager calls recover on the resource only once. Only applies to data sources that use an XA driver. Use this setting to work around specific problems with JDBC XA drivers. Default: `false`.
`need-tx-ctx-on-close`	`boolean`	Specifies whether the XA driver requires a distributed transaction context when closing various JDBC objects (result sets, statements, connections, and so forth). Only applies to connection pools that use an XA driver. When enabled, SQL exceptions that are thrown while closing the JDBC objects without a transaction context will be suppressed. Use this setting to work around specific problems with JDBC XA drivers. Default: `false`.

The following example shows how to use the `xa-params` element in the Oracle Event Processing server configuration file:

```
<data-source>
    <name>orads</name>
    <xa-params>
        <keep-xa-conn-till-tx-complete>true</keep-xa-conn-till-tx-complete>
    </xa-params>
    <driver-params>
        <url>jdbc:oracle:thin:@localhost:1521:ce102</url>
        <driver-name>oracle.jdbc.OracleDriver</driver-name>
        <properties>
            <element>
                <name>user</name>
                <value>wlevs</value>
            </element>
            <element>
                <name>password</name>
                <value>wlevs</value>
```

```
            </element>
        </properties>
    </driver-params>
    <connection-pool-params>
        <initial-capacity>5</initial-capacity>
        <max-capacity>10</max-capacity>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <test-frequency-seconds>5</test-frequency-seconds>
    </connection-pool-params>
    <data-source-params>
        <jndi-names>
            <element>orads</element>
        </jndi-names>
        <global-transactions-protocol>None</global-transactions-protocol>
    </data-source-params>
</data-source>
```

8

High Availability Schema

This chapter provides a reference to the `ocep_cluster_config.xsd` schema. The `ocep_cluster_config.xsd` file describes the high availability clustering configuration elements and attributes

All of the high availability adapters are based on the basic adapter configuration and add elements specific to their type of adapter. See Component Element Hierarchies for the basic adapter elements. This section covers the elements specific to high availability adapters.

This chapter includes the following sections:

8.1 ha-buffering-adapter

Use the `ha-buffering-adapter` element to create a high availability buffering adapter. This element has the following child elements:

- warm-up-window-length
- window-length

The following example shows how to use these elements in a configuration file:

```
<ha:ha-buffering-adapter >
  <name>mySlidingWindowingAdapter</name>
  <window-length>15000</window-length>
  <warm-up-window-length units="minutes">6</warm-up-window-length>
</ha:ha-buffering-adapter >
```

8.2 ha-broadcast-adapter

Use the `ha-broadcast-adapter` to create a high availability broadcast adapter. The element has the following child elements:

- trimming-interval
- warm-up-window-length

The following example shows how to use the trimming-interval element in a configuration file:

```
<ha:ha-broadcast-adapter>
  <name>myBroadcastAdapter</name>
  <trimming-interval units="events">10</trimming-interval>
  <warm-up-window-length units="minutes">6</warm-up-window-length>
</ha:ha-broadcast-adapter>
```

8.3 ha-correlating-adapter

Use the `ha-correlating-adapter` to create a high availability correlating adapter. This element has the following child elements:

- batch-size
- heartbeat
- fail-over-delay
- trimming-interval
- warm-up-window-length
- window-length

The following example shows how to use the `fail-over-delay` element in a configuration file:

```
<ha:ha-correlating-adapter>
  <name>myHaBroadcastAdapter</name>
  <fail-over-delay>2000</fail-over-delay>
</ha:ha-correlating-adapter>
```

8.4 ha-inbound-adapter

Use the `ha-inbound-adapter` to create a high availability input adapter. This element has the following child elements:

- batch-size
- heartbeat

The following example shows how to use the `heartbeat` and `batch-size` elements in a configuration file:

```
<ha:ha-inbound-adapter>
  <name>myHaInputAdapter</name>
  <heartbeat units="millis">1000</heartbeat>
  <batch-size>10</batch-size>
</ha:ha-inbound-adapter>
```

8.5 batch-size

`batch-size`: An integer value that indicates the number of events to send to the secondary server with the same time stamp. By default, batching is disabled (value 0).

8.6 fail-over-delay

`fail-over-delay`: The time in milliseconds that a secondary server waits to received a correlating message before considering the primary to have failed.

8.7 heartbeat

`heartbeat`: The value (n) for the heartbeat time out on this adapter. Oracle Event Processing generates a heartbeat after `n` time units of time go by without any event being generated on this adapter. The default time unit is `nanoseconds`.

8.8 trimming-interval

`trimming-interval`: The interval at which Oracle Event Processing trims events from a secondary server buffer. Units are either `events` or `milliseconds`.

8.9 warm-up-window-length

`warm-up-window-length`: The length of the warm-up window. Units can be in seconds (default) or minutes. The warm-up window is the length of time that a secondary server needs before it begins to produce valid output events after joining the cluster. Set the warm-up window long enough to allow the input windows of all Oracle CQL queries to be fully populated. By default, the length is 0 meaning no warm-up window.

8.10 window-length

`window-length`: The length of the warm-up window. Units can be in seconds (default), or minutes and type `Long` with units (seconds or minutes). The warm-up window is the length of time that a secondary server needs before it begins to produce valid output events after joining the cluster. Set the warm-up window length long enough to allow the input windows of all CQL queries to be fully populated. By default, the length is 0 seconds meaning there is no warm-up window.

9

Data Cartridge Schema

This chapter provides the schema reference information for the Oracle Event Processing data cartridge schema files. You can access these files in your Oracle Event Processing installation at `/Oracle/Middleware/my_osa/osa/xsd`.

This chapter includes the following sections:

9.1 ocep_jdbc_context_config.xsd

The `ocep_jdbc_context_config.xsd` file defines configuration settings for a JDBC data cartridge. A JDBC cartridge configuration has the following elements and attributes.

- `data-source`: The name of the database connection between the server and the database.
- `function`: The JDBC function you want to call to query a database table. A JDBC Cartridge function is a `TABLE` function that returns a collection.
- `param`: A set of required `name` and `type` pairs that define parameter values that are passed to the `function` for a database query. The `type` value is an Oracle CQL simple type.
- `return-component-type`: A simple Oracle CQL data type such as `INT` or `FLOAT`, a complex type such as a POJO bean name, or an extensible type defined by a cartridge. The `return-component-type` represents the component type of the `collection-type` returned by the JDBC cartridge `TABLE` function.
- `sql`: The SQL query that defines the `function`. Precede references to the parameters declared in the `param` element with a colon (:).

The following entries in the configuration file show how to use these elements and their attributes to create a JDBC cartridge configuration with an SQL statement.

```
<jc:jdbc-ctx>
    <name>JdbcCartridgeOne</name>
    <data-source>StockDS</data-source>
    <function name="getDetailsByOrderIdName">
        <param name="inpOrderId" type="int" />
        <param name="inpName" type="char" />
        <return-component-type>
```

```
            com.oracle.cep.example.jdbc_cartridge.RetEvent
        </return-component-type>
        <sql><![CDATA[
            SELECT
                Employee.empName as employeeName,
                Employee.empEmail as employeeEmail,
                OrderDetails.description as description
            FROM
                 PlacedOrders, OrderDetails , Employee
            WHERE
                PlacedOrders.empId = Employee.empId AND
                PlacedOrders.orderId = OrderDetails.orderId AND
                Employee.empName = :inpName AND
                PlacedOrders.orderId = :inpOrderId
        ></sql>
    </function>
</jc:jdbc-ctx>
```

9.2 ocep-jdbc.xsd

The `ocep-jdbc.xsd` file defines assembly settings for a JDBC data cartridge. A JDBC cartridge assembly configuration has the has the `jdbc-context` element. The `jdbc-context` element declares the Oracle Event Processing JDBC cartridge context.

The following entry in the assembly file sets the JDBC cartridge context to `JdbcCartridgeOne`.

```
<jdbc:jdbc-context id="JdbcCartridgeOne"/>
```

9.3 ocep-spatial.xsd

The `ocep-spatial.xsd` file defines assembly settings for a spatial `context`. Use a spatial `context` to specify the coordinate system you want to use and the associated attributes, as follows.

A spatial `context` has the following attributes:

- `cartesian`: Specify `true` to use cartesian coordinates. Default is `false` which defaults to Geodetic (WGS84) coordinates.
- `srid`: Identifies the coordinate system. The default is `LAT_LNG_WGS84_SRID` for the Geodetic (WGS84) coordinate system. Specify `CARTESIAN` for the Cartesian coordinate system.
- `sma`: Specify a `double` value that defines the semi-major axis parameter. The semi-major axis parameter value is used for buffering and projection. The default value is `6378137.0`.
- `rof`: Specify a `double` value that defines the flattening parameter reciprocal. The flattening parameter reciprocal is used for buffering and projection. The default value is `298.257223563`.
- `tolerance`: Specify a `double` value that defines the minimum distance to be ignored in geometric operation including buffering. The default value is `0.000000001`.
- `anyinteract-tolerance`: Specify a `double` value that defines the default tolerance for `ANYINTERACT` Oracle Spatial geometric relation operator. The default value is `0.0000005`.

The following entry in the assembly file shows a spatial context configuration:

```
<spatial:context id="my-spatial-context" anyinteract-tolerance="0.0000005"
  rof="298.257223563" sma="6378137.0" srid="LAT_LNG_WGS84_SRID"
  tolerance="0.000000001"/>
```

9.4 ocep-hadoop.xsd

The `ocep-hadoop.xsd` file defines assembly settings for a `file` that stores event information that resides in the Hadoop file system or in a local file system. The `file` element has the following attributes:

- `event-type`: Specify an event type that defines the schema of the data to be stored in the Hadoop HDFS or local FS.
- `path`: Specify a file path in the Hadoop HDFS or local FS where you want Oracle Event Processing to store the event data. A path string is absolute if it begins with a slash.
- `separator`: Specify the file path separator. The path separator relates to how the data is separated within the referenced file. The path points to a CSV file, for example, and the separator specifies that the fields are separated with a comma (,).
- `operation-timeout`: Specify how long to wait in milliseconds for the operation to complete. If the operation does not complete within the specified amount of time, the operation returns with an empty result and then continues normally.

The following entry in the assembly file shows a Hadoop configuration:

```
<hadoop:file id="hadoop" event-type="OracleL2StockTick"
  path="/my_hadoop/hadoop/hadoop_file" separator=","/>
```

9.5 ocep-nosql.xsd

The `ocep-nosqldb.xsd` file defines assembly settings for a data `store` that contains event information that resides in NoSQLDB. The `store` element has the following attributes:

- `event-type`: Specify an event type that defines the schema of the data to be stored in NoSQLDB. The `event-type` must be a Java-based event type that implements the `java.io.Serializable` interface.Class-loader of application that provides this event-type is used when deserializing the Java object put in this store.
- `store-name`: The name of the Oracle NoSQLDB store.
- `store-locations`: A space-separated list of the locations of the NoSQLDB nodes in the form of `domain:port`. For example, `localhost:5000`. Oracle Event Processing uses the locations to connect to a NoSQLDB server. If a server is not available, Oracle Event Processing uses the next location in the list.

The following entry in the assembly file shows a NoSQLDB configuration:

```
<nosqldb:store id="my_nosqldb" event-type="TradeEvent2"
  store-name="my_nosql_store" store-locations="localhost:5000"/>
```

10

Metatype Schema

The `ocep_metatype_config.xsd` file describes the configuration elements and attributes for binding an object class to its attribute definition.

This appendix includes the following sections:

10.1 binding

Use the `binding` element to provide a `String` value that specifies the implementation binding of an object class and an attribute definition. For example, the value of `jmx:EventChannel` in an object class definition binds it to a JMX MBean of type `EventChannel`. You do not need to include a protocol when you use this `binding` element because the protocol is clear from the containing object class definition.

This element has no child elements or attributes.

10.2 multi-valued

Use the `multi-valued` element to specify a `Boolean` value that indicates whether to use the contained attribute definitions collectively to form a single multivalue attribute. For example, in CQL Processor parameter bindings.

This element has no child elements or attributes.