Oracle® Sensor Edge Server

Administrator's Guide

10*g* Release 2 (10.1.2)

Part No. B14455-01

December 2004

ORACLE®

Oracle Sensor Edge Server Administrator's Guide, 10*g* Release 2 (10.1.2)

Part No. B14455-01

Contents

2 Managing Drivers

3 Managing Filters

4 Managing Extensions

A Sample edgeserver.xml File

B Troubleshooting

Glossary

Index

Send Us Your Comments

Oracle Sensor Edge Server Administrator's Guide, 10*g* Release 2 (10.1.2)

Part No. B14455-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Application Server Documentation
  500 Oracle Parkway, M/S 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

The Oracle Sensor Edge Server Administrator's Guide describes how to configure and manage the Oracle Sensor Edge Server. This guide includes the following chapters.

Chapter 3, "Managing Filters"

Describes how to configure the filters and filter instances used by the drivers and devices of the Oracle Sensor Edge Server.

Chapter 4, "Managing Extensions"

Describes the Extension Archive file format for packaging custom drivers, dispatchers, and filters and how to upload Extension Archive files to the Oracle Sensor Edge Server.

Appendix A, "Sample edgeserver.xml File"

Provides a sample of the configuration file for the Oracle Sensor Edge Server.

Appendix B, "Troubleshooting"

Provides solutions to problems encountered when running the Oracle Sensor Edge Server.

Related Documents

For more information, see the following manuals:

- *Oracle Application Server Wireless Developer's Guide*
- *Oracle Application Server Wireless API Reference*
- *Oracle Application Server Wireless Administrator's Guide*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

Configuring the Oracle Sensor Edge Server

This chapter, through the following sections, describes how to configure the Oracle Sensor Edge Server to receive, filter, and dispatch data.

- Section 1.1, "Overview of the Oracle Sensor Edge Server"
- Section 1.2, "Overview of the Oracle Sensor Edge Server Configuration File"
- Section 1.3, "Starting and Stopping the Oracle Sensor Edge Server"
- Section 1.4, "Configuring the Dispatchers for an Oracle Sensor Edge Server"
- Section 1.5, "Connecting Readers and Sensors"
- Section 1.6, "Enabling Devices to Filter Events"

1.1 Overview of the Oracle Sensor Edge Server

The Oracle Sensor Edge Server enables enterprises to incorporate information received from sensors devices into their I.T. infrastructure and business applications by receiving event data from sensor devices and then normalizing this data by putting it in a common data format and filtering out excess information. The event data, which is now a normalized event message, is then sent to database applications.

1.1.1 Deploying Drivers, Filters and Dispatchers to the Oracle Sensor Edge Server

To receive and process event data from sensor devices, the Oracle Edge Sesor Server uses driver and filter objects to receive, read events and then process the event data. The Oracle Edge Sensor Server then sends the processed event data on to applications using a dispatcher. Depending on the configuration of the Oracle Sensor Edge Server's dispatcher, an Oracle Sensor Edge Server client application receives event messages through database streams, JMS (Java Message Service), Web Services (SOAP), or HTTP. The payload of the message is always an event. For more information on dispatching events, refer to Section 1.4.

The drivers provided out of the box support the 9RE-001 by Alien Technology, the Penn, Delaware, and PCMCIA readers manufactured by Intermec and the PHE-3FB PC-Controlled Light indicator manufactured by Patlite Corporation. For more information on drivers, refer to Chapter 2, "Managing Drivers". For filtering event data, the Oracle Sensor Edge Server provides filters that perform not only such functions as device monitoring, event tracking, and data cleansing by blocking redundant events, but can also send events that signal when new containers or pallets enter or exit the field or gateway of a device reader, or enable you to see all of the tag IDs of items held in a container or on a pallet. For more information on the filters provided by the Oracle Sensor Edge Server, refer to Chapter 3, "Managing Filters".

You can change or further enhance the capabilities of the Oracle Sensor Edge Server by adding extensions: custom-built drivers, filters and dispatchers. You add the drivers, filters and dispatchers by downloading an Extension Archive file (a JAR file containing all of the class files as well as other related files) to the repository. For more information, refer to Chapter 4, "Managing Extensions".

For the Oracle Sensor Server to use extensions, however, you must create instances of these objects. For example, for a sensor to use a driver, you must create an instance of the driver object called a device instance. Similarly, for device instance to utilize a filter, you must create a filter instance of the filter object. When you assign a dispatcher as the current dispatcher used by the Oracle Sensor Edge Server, you create an instance of a dispatcher. For more information on the components of the Oracle Sensor Edge Server, refer to the *Oracle Application Server Wireless Developer's Guide*

Note: If you modify the devices, device groups, or current dispatchers (or modify the devices, device groups, or dispatchers in the template section of `edgeserver.xml`), then you must stop and restart the Oracle Sensor Edge Server so that any changes can take effect. For more information on starting and stopping the Oracle Sensor Edge Server, refer to Section 1.3.

1.1.2 Overview of Events

The Event type represents a message sent from one component of the Oracle Sensor Edge Server to another. These event messages are specific to each type of driver or filter. The Event type is divided into the following sections:

- Header
- Type Info
- Payload

Header

The Header sections includes the routing headers and the message headers, which contain fields that designate the delivery of an event message. The routing fields include `<sourceName>` and `<correlationID>`. The message headers include the `<siteName>` and `<deviceName>` fields.

- `<sourceName>`

  This field identifies the originator of the event. This is an optional field, one set by the client.

- `<correlationId>`

  The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message response sent back by the client will have the same ID. This is an optional field.

- `<siteName>`

  The site that originally generated the message.

- `<deviceName>`

  The name of the device or application that generates the event.

- `<time>`

  The date and time when the observation or message was created.

Type Info

The Type Info section contains the formatting information for the payload: the type and subtype of the event.

- `<type>`

  The number value that corresponds to the type of event. Table 1–1 describes the values of the `<type>` field. The *Oracle Application Server Wireless Developer's Guide* describes the values for the `<type>` field in further detail.

Table 1–1 Value Ranges for the <type> Field

Range	Message Type
0-99	System messages. The range of values includes: ■ 0 -- Unknown A value of 0 represents a bad event or a system internal event. ■ 1 -- Message Event A confirmation message, usually the return result code of a corresponding instruction.
100-199	Instructions or commands. The range of values includes: ■ 100 -- General Instructions General Instructions for controlling devices. ■ 101 -- RFID Instruction Instructions to RFID devices. ■ 102 -- Printer Instructions to printers. ■ 103 -- Lightstack
200-299	Observations. The range of values includes: ■ 200 -- RFID Observation The message is an RFID observation ■ 201 -- RTLS ■ 202 -- Physical Contact ■ 203 -- Temperature ■ 204 -- Humidity ■ 205 -- Weight ■ 206 -- Tampering ■ 207 -- Audio ■ 208 -- Message Board
500-599	Custom messages

- `<subtype>`

  The number value for the subtype. For more information on Instruction Events, refer to the *Oracle Application Server Wireless Developer's Guide*

Payload

The Payload section, which is the event-specific section with the following fields:

- `<id>`

  The text value of this field identifies a tag (that is, a read or target) to an event instruction.

- `<data>`

  The tag data. This is an optional field.

1.2 Overview of the Oracle Sensor Edge Server Configuration File

You configure the Oracle Sensor Edge Server by editing its configuration file, `edgeserver.xml`. This file defines all of the parameters and runtime settings for the Oracle Sensor Edge Server. When the Oracle Sensor Edge Server first starts, it reads the file to setup all of the runtime data and loads all of the necessary components and extensions. `edgeserver.xml` is located at

`ORACLE_HOME/edge/config/edgeserver.xml`

Note: The Oracle Sensor Edge Server also stores its cache file at this location.

Example 1–1 illustrates the basic structure of this XML file.

Example 1–1 Structure of edgeserver.xml

```
<EdgeServer>
   <DispatcherList>
     <Dispatcher>...</Dispatcher>
   </DispatcherList>
   <DriverList>
      <Driver>...</Driver>
   </DriverList>
   <FilterList>
      <Filter>...</Filter>
   </FilterList>
   <CurrentDispatcher>
      <DispatcherName> ...</DispatcherName>
   </CurrentDispatcher>
   <DeviceGroups>
      <DeviceGroup>
         <DeviceList>
            <Device>...</Device>
         </DeviceList>
      </DeviceGroup>
   </DeviceGroups>
</EdgeServer>
```

Note: Refer to Appendix A for an example of the entire `edgeserver.xml` file.

1.2.1 The General Settings and Parameters for the Oracle Sensor Edge Server

The file's root element, <EdgeServer> encompasses high-level configurations (described in Table 1–2) and contains elements that enable you to configure not only the dispatchers, filters, and drivers available to the Oracle Sensor Edge Server, but the instances of these objects that enable the Oracle Sensor Edge Server to receive and process events.

Table 1–2 General Configuration for the Oracle Sensor Edge Server

Element	Description
<Name>	The name of the Oracle Sensor Edge Server.
<SiteName>	The name of the site where the Oracle Sensor Edge Server resides.
<DispatcherMode>	The mode for the internal queue. Values include: ■ Persist (default mode) -- Persists all events before dispatching. ■ Normal -- Persists only when there is an error while dispatching events. ■ Memory -- No persistence; the events are stored in memory before they are dispatched.
<IsExtensionMonitorEnabled>	When set to *true* (enabled), the extension monitor can add extensions at runtime.
<LogLevel>	Notify, Warning, Error

1.2.2 Available Dispatchers, Filters, and Drivers

The <DispatcherList>, <DriverList> and <FilterList> elements comprise the template section of the configuration file, where you define the filters, dispatchers and drivers that are available to the Oracle Sensor Edge Server.

Dispatchers, filters and drivers are extensions, downloadable components that add functionality to the Oracle Sensor Edge Server. By themselves, these extensions are not active components; they are not executed. For the Oracle Sensor Edge Server to use these components, you must instantiate an extension by creating an instance of it. For example, to connect a driver to a reader, you must create a device instance of the driver extension. Table 1–3 describes the components for extensions that can be configured. The instances of these extensions (described in Section 1.2.4.1) are called devices and filter instances. The instance of a dispatcher is the current dispatcher.

Table 1–3 Structure of the Extensions

Component	Description
ClassName	The name of the class that implements the specific extension interface. This class is loaded first when an instance is created from an extension.
Description	A text description of the extension or instance.
Name	A unique name for the extension or instance.
Parameters	A list of parameters specific to the extension or instance.

Defining Parameters for Extensions and Instances

Both extensions and their instances contain dynamic lists of parameter settings which are defined within the <Parameters> elements of the configuration file. These parameters are specific to the type of extension and can be configured for each instance. For extensions, the <Parameters> elements and its child, <Parameter>

tell the Oracle Sensor Edge Server the default values of the parameters and which parameters (if any) are exposed.

<Parameter> requires the following fields:

- id

  An attribute with a unique identifier. Some of the tags in edgeserver.xml (including <Parameters>) require a unique identifier so that they can be referenced by other tags. The identifier is defined in the id attribute and must be unique across the configuration file (although they do not have to be consecutive). When adding new elements that do not have references, you do not have to manually add the value of the reference id attribute; the Oracle Sensor Edge Server automatically assigns a value once it starts.

  Note: Entering an incorrect reference id might lead to incorrect behavior.

- name

  The name of the attribute, unique within the <Parameters> tag.

- defaultValue

  The default value for the parameter. This is used as the initial value in any UI or when first creating the object

- description

  Text that describes the value (mostly for the user interface).

- encrypted

  Indicates whether the value for the parameter should be encrypted so that the value does not have to be stored in clear-text format.

- isClearText

  Enables the default value (and the value for the parameter instance) to be reset to clear-text format. If the encrypted parameter is set to true, then the clear-text format is read and then set to encrypted format the next time the Oracle Sensor Edge Server starts.

- required

  Indicates whether the parameter value is required.

- valueType

  The type attribute specifics the type to use for the value. It could be

  - string for string values
  - int for integer values
  - boolean for true or false values
  - double for a double-precision number

1.2.2.1 Dispatchers

The <DispatcherList> element specifies all of the dispatcher extensions that are uploaded to the system and are available to the Oracle Sensor Edge Server. It contains an array of <Dispatcher>, which defines an uploaded dispatcher. There can only be

one <DispatcherList> in the <EdgeServer> node. Each <Dispatcher> is an extension and is in the format described in Table 1–3.

1.2.2.2 Drivers

The <DriverList> element defines all of the uploaded driver extensions. All supported drivers must be listed within this element. The <DriverList> contains a list of <Driver> elements. For the format of the <Driver> element, refer to Table 1–3. Only one <DriverList> can be defined in the system.

1.2.2.3 Filters

The <FilterList> section lists all of the filters available to the Oracle Sensor Edge Server. Each filter is defined within <Filter>.

1.2.3 The Current Dispatcher Method for the Oracle Sensor Edge Server

<CurrentDispatcher> defines which dispatcher the Oracle Sensor Edge Server uses as well as the dispatcher's parameters. This is the only instance object that is not defined in the <DeviceGroup> section (described in Section 1.2.4). An Oracle Sensor Edge Server can use only one dispatcher at a time. Even if you have only one dispatcher defined within <DispatcherList>, the Oracle Sensor Edge Server can use it only if it is defined within the <CurrentDispatcher> element. After you assign a dispatcher as current, you must stop and then start the Oracle Sensor Edge Server. For more information setting the current dispatcher method in edgeserver.xml, refer to Section 1.4.1. For information on starting and stopping the Oracle Sensor Edge Server, refer to Section 1.3.

1.2.4 Filters, Dispatchers and Devices Used by the Oracle Sensor Edge Server

Device groups are similar to the root or top-level directories of a file system. Because all devices (the instances of drivers) and filter instances must be part of a device group, you must create one or many device groups before you can connect devices and set up filters. You can group devices in terms of management if you want to treat them as a logical unit to manage, or you can group them by the type of filtering that they perform. For example, if you group devices by cross-reader filtering, then you create a group of related devices and then attach filters to that group.

The <DeviceGroups> section of the configuration file defines all the device groups in the system. All of the instances of devices and filters are defined within this element. The <DeviceGroups> element contains an array of <DeviceGroup> elements. Each <DeviceGroup> tag contains the following:

- <Device>

  One of many <Device> elements that can be defined in a device group. Each <Device> defines a device instance.

- <EventCollectWaitTime>

  The interval for the group to pull events out of the filtered window of events, in milliseconds.

- <FilterInsts>

  Defines the list of group filter instances

- <IsDefault>

  Set to *false* for all user groups

- <Name>

  The name of the device group

For more information on defining devices, refer to Section 1.5. For more information on creating filter instances, refer to Section 3.3.1.

1.2.4.1 Setting the Parameters for Instances

Instance configurations include:

Instance Details

The name of the instance.

Instance Type

The type of the instance (device, filter instance, or current dispatcher) as well as a reference to the declaration of the component in the template section of edgeserver.xml. In Example 1–2, the <Extension> tag for the Itermec Device has the reference attribute set to 35 and refers to <Driver id="35">, the IntermecDevice driver defined in the <DriverList> element of edgeserver.xml.

Example 1–2 Extension Reference to a Driver

```
<Name>Intermec Device</Name>
<DriverName>IntermecDevice</DriverName>
<Extension Reference="35">/
```

Parameter Configurations

The parameters required by the instance. These parameters must also point to the declaration of the parameters for the component defined in the template section of edgeserver.xml.

The tag <ParameterInsts>, defines the parameters for an instance. The individual parameters are defined with the tag, <ParameterInst> (described in Example 1–3).

<ParameterInst> requires the following elements:

- <Name>

  The name of the parameter. The value must match the name attribute of the corresponding <Parameter> of the extension.

- <ParameterMetaData>

  The reference attribute must be set to the id of the corresponding parameter in the extension (which is defined within the <DispatcherList>, <DriverList>, or <FilterList> elements.)

- <Value>

  The value for the parameter for this instance.

Example 1–3 The <ParameterInst> Element

```
<ParameterInst id="79">
        <Name>password</Name>
        <ParameterMetaData reference="8"/>
        <Value>oracle</Value>
</ParameterInst>
```

1.3 Starting and Stopping the Oracle Sensor Edge Server

The Oracle Sensor Edge Server starts by default when you start the Oracle Application Server Containers for J2EE instance (OC4J instance) and stops once you have stopped the OC4J instance. You can also start the Oracle Edge Sensor Server by running the `runSensorEdgeServer.bat` or `runSensorEdgeServer.sh` commands. If you modify the configuration file, then you must stop and restart the Oracle Sensor Edge Server. For information on starting and stopping the Oracle Sensor Edge Server after adding extensions, refer to Section 4.3.

Note: You must install JDK 1.4. or higher to run the Oracle Sensor Edge Server.

1.3.1 Starting the Oracle Sensor Edge Server by Starting an OC4J Instance

To start the OC4J instance from a command line:

1. Start a command shell (on Windows, for example, use `cmd.exe`).
2. Navigate to `ORACLE_HOME/j2ee/home`.
3. Execute `java -jar oc4j.jar` to start the OC4J instance.

A message similar to the one illustrated in Example 1–4 displays if the Oracle Sensor Edge Server has been installed properly.

Example 1–4 Confirmation Message

```
04/11/17 20:36:07 Extension upload monitoring is not enabled...
04/11/17 20:36:07 EdgeServer: finished initialization
04/11/17 20:36:07 Oracle Application Server Containers for J2EE 10g (10.1.2.0.0)
initialized
```

1.3.2 Stopping Oracle Sensor Edge Server by Stopping the OC4J Instance

You can stop the Oracle Sensor Edge Server by using the OC4J administrative tools or you stop the OC4J instance itself from the command line. In either case, the Oracle Sensor Edge Server is notified and shuts down gracefully. To stop the OC4J instance from the command-line, send a kill signal to the OC4J instance. For example, on Windows, click Ctrl+C to kill the instance.

Tip: Refer to the *Containers for J2EE User's Guide* for information on other methods of starting or stopping an OC4J instance.

1.3.3 Starting the Oracle Edge Sensor Server From the Command Line

In addition to starting the Oracle Sensor Edge Server by starting an OC4J instance, you can also start the Oracle Sensor Edge Server by running the following:

On Windows:

`ORACLE_HOME\edge\runSensorEdgeServer.bat`

On UNIX:

`ORACLE_HOME/edge/runSensorEdgeServer.sh`

If you do not provide the argument specifying the `ORACLE_HOME`, then the system prompts you with the following message:

```
Must provide the  for this Sensor Edge Server.Usage:
runSensorEdgeServer.bat ORACLE_HOMEORACLE_HOME
```

If you provided the ORACLE_HOME argument, then a message simular to the one described in Example 1–5 appears.

Example 1–5 Confirmation Message

```
04/11/17 20:36:07 Extension upload monitoring is not enabled...
04/11/17 20:36:07 EdgeServer: finished initialization
04/11/17 20:36:07 Oracle Application Server Containers for J2EE 10g (10.1.2.0.0)
initialized
```

1.4 Configuring the Dispatchers for an Oracle Sensor Edge Server

The <DispatcherList> element of the configuration file (described in Section 1.2.2) defines the dispatchers that have been uploaded to the Oracle Sensor Edge Server. For the Oracle Sensor Edge Server to use a dispatcher, you must create a dispatcher, an instance of an available dispatcher, by defining the <CurrentDispatcher> element (described in Section 1.4.1). An Oracle Sensor Edge Server can use only one dispatcher. By defining the elements within <CurrentDispatcher> you can direct the Oracle Sensor Edge Server to send event messages using Oracle Streams, Oracle's Java Message Service provider (OC4J JMS) remote Web Services or HTTP (to a client application).

1.4.1 Setting the Current Dispatcher Used by the Oracle Sensor Edge Server

To set the current dispatcher for the Oracle Sensor Edge Server:

1. Select from among the dispatchers defined within <DispatcherList>.
2. In <CurrentDispatcher>, change the element content of <DispatcherName> to the name of the dispatcher, which could be any one of the following:
   - Streams Dispatcher (Section 1.4.1.1)
   - JMS Dispatcher (Section 1.4.1.2)
   - HTTP Dispatcher (Section 1.4.1.3.
   - Web Services (Section 1.4.1.4)
   - NullDispatcher (Section 1.4.1.5).
3. Compare the definition of <Dispatcher> within <DispatcherList> to ensure that each <ParameterInst> within <CurrentDispatcher> corresponds with each <Parameter> defined within <Dispatcher>.
4. Save edgeserver.xml and restart the Oracle Sensor Edge Server. The server then loads the selected dispatcher upon startup.

Example 1–6 illustrates configuring the <CurrentDispatcher> elements for the Streams dispatcher.

Example 1–6 Configuring <CurrentDispatcher> in edgeserver.xml

```
<CurrentDispatcher id="76">
   <DispatcherName>StreamsDispatcher</DispatcherName>
       <Extension class="Dispatcher" reference="5"/>
       <Name>StreamsDispatcher</Name>
       <NeedReload>false</NeedReload>
   <ParameterInsts id="77">
```

```
      <ParameterInst id="78">
         <Name>username</Name>
         <ParameterMetaData reference="7"/>
         <Value>edge</Value>
      </ParameterInst>
      <ParameterInst id="79">
         <Name>password</Name>
         <ParameterMetaData reference="8"/>
         <Value>oracle</Value>
      </ParameterInst>
      <ParameterInst id="80">
         <Name>url</Name>
         <ParameterMetaData reference="9"/>
         <Value>jdbc:oracle:thin:@(description=(address=(host=soclxs3db02)
                     (protocol=tcp)(port=9105))(connect_data=(sid=PRJ1)))</Value>
      </ParameterInst>
   </ParameterInsts>
</CurrentDispatcher>
```

1.4.1.1 Configuring the Oracle Sensor Edge Server to Use Oracle Streams

Configuring the dispatcher as Oracle Streams and Advanced Queuing enables you to control how the dispatcher retrieves and distributes event messages. Unlike the OC4J JMS, Web Services, or HTTP dispatcher options, event messages dispatched using the Oracle Streams dispatcher do not have to be relayed directly to an entry point. The Oracle Streams dispatcher supports rule-based process and agent technologies. In addition, the Oracle Streams dispatcher only supports UTF-8 encoding.

Tips:

- Because Oracle Streams enables the propagation and management of data, transactions, and events in a data stream on one -- or across many -- databases, this dispatcher option provides the greatest flexibility of the seeded dispatcher options.
- The Oracle Streams dispatcher requires JDK 1.4.x.

Once the event messages are captured and placed into the staging queue, the event message data can be processed through the Rules Evaluation Job, which takes event messages from the staging queue and then compares them to the Oracle Sensor Edge Server rule set. Each rule has an action, which is executed if the rule applies. These actions include a PL/SQL callback for propagating the event message to other queues which could then be consumed by other applications. For more informationon Oracle Sensor Edge Server Rule Set, refer to the *Oracle Application Server Wireless Developer's Guide*

Event messages are data that is deposited to a staging area (an internal queue). This data, in turn, can be aggregated in different ways for different client devices and applications (the consumers of the event messages). Using Oracle Streams as the dispatcher, the Data and Event layer of the database, not the Oracle Sensor Edge Server or applications, determines what an event is and when it is generated. The Data and Event layer provides a rule-based process that determines the appropriate event message for each client device or application.

In addition to the these rule-based actions, the Rule Evaluation Job invokes applications by calling the Sensor Data Hub (SDH), which receives sensor data from the Oracle Sensor Edge Server or from other sources. The SDH includes the Sensor Data Archive, a set of archive tables that store all of the filtered sensor events in the system.

To configure the Streams dispatcher:

1. Within <CurrentDispatcher>, change the element content of <DispatcherName> to <StreamsDispatcher>.
2. Compare the attributes defined for the following connection information for the Streams staging area in the <ParameterInst> elements to those defined in the <Parameter> elements for the Streams Dispatcher:
   - url

     The JDBC connection string to the database where the staging area resides. By default, this value is set to the OracleAS infrastructure database. Enter the URL to an application database providing optimal access and archiving of events and observations. The URL depends upon the driver type. For example, for a thin driver, enter

     ```
     jdbc:oracle:thin@(description=(address=(host=<hostname>)<p
     rotocol=tcp)(port=<port>))(connect_data=(sid=<sid>)))
     ```

     where <hostname> is the host name or IP of the database server, <port> is the port number for the listener (1521, by default) and <sid> is the service id of the instance.
   - username

     The name of the user of the database where the staging area resides.
   - password

     The password to the user of the database where the staging area resides.
3. Save the edgeserver.xml and restart the Oracle Sensor Edge Server. The server loads the Oracle Streams dispatcher upon startup.

> **Note:** Applications requiring raw, unfiltered event data that has not been processed by the rules can connect to the staging area using either AQ notification or JMS.

1.4.1.1.1 Post-Installation Steps for the Streams Dispatcher

If you configure database streams as the event dispatcher method, you must perform the following post-installation steps:

1. Set up a database instance to run the Oracle Streams disptacher. This database instance holds the staging area for the Oracle Streams dispatcher, as well as its rules, queues, Sensor Data Archive (SDH), and other related data. You can use either the Standard or Enterprise Edition of the database as long as it is Version 9.2 or higher. Be sure to note the connect string and password. After you set up the database, you must set up the schema used by the Oracle Streams dispatcher.

   > **Note:** if the database instance is on a separate box, you must edit the ORACLE_HOME/network/admin/tnsnames.ora file on the machine running the Oracle Sensor Edge Server or add the GDN to it.

2. Enter the database password.
3. Navigate to the SQL directory (such as ORACLE_HOME/edge/sql).
4. Use SQL*Plus to connect to the database (using the system account).

5. Run `create_edg_user.sql` to create a user permission.
6. Enter a password for the user. You can enter any password. Be sure to note the password.
7. Disconnect as system.
8. Reconnect as the user of the Oracle Sensor Edge Server using the password that you entered in Step 6.
9. Run `edg_create_streams.sql`.
10. Install the Sensor Data Hub (SDH) and then run `edg_create_sdh.sql`(as the EDGE user). The database includes a user called EDGE after you execute these scripts. All of the required schemas and data are created under that user. Background jobs start automatically.

1.4.1.2 Configuring the Dispatcher to Send Messages Through OC4J JMS

OC4J JMS (OracleAS Containers for J2EE and Java Message Service), which is compatible with J2EE 1.3, is a Java Message Service based on Advanced Queuing (AQ) that provides guaranteed message delivery with auditing.

Note: In the current release, the OC4J JMS dispatcher configuration (that is, the JMS dispatcher) cannot send messages back to sensor devices. Only Oracle Streams supports this functionality.

To enable event message dispatching using OC4J JMS, you must configure a JMS topic queue called `edgeTopic` to which the dispatcher posts new event messages. You must also specify a topic connection factory, `edgeEventsConnectionFactory`. To enable the Oracle Sensor Edge Server components to access this topic, you must configure the `jms.xml` file under the OC4J container. For more information on configuring the JMS queue, refer to *Oracle Application Server Containers for J2EE Services Guide*.

Note: Upon startup, the JMS dispatcher looks for the `edgeTopic` queue using the JNDI (Java Naming Directory Interface) service implemented by the OC4J container. If the dispatcher cannot locate `edgeTopic`, it returns an error and then exits.

1.4.1.2.1 Configuring jms.xml Under the OC4J Container To configure the `jms.xml` file under the OC4J container:

1. Note all of the OC4J JMS Server information, such as hostname, JMS username (typically *admin*), JMS password, and the RMI port information.
2. Navigate to `ORACLE_HOME/j2ee/home/config`.
3. Edit the `jms.xml` file by adding the following lines:

```
<topic-connection-factory
          host="<hostname>"
          location="jms/TopicConnectionFactory"
          password="<password>"
          port="<jmsport>"
          username="admin" />
```

```
<topic name="jms/edgeTopic" location="jms/edgeTopic">

         <description>The edge server event
topic</description>

</topic>
```

4. Save `jms.xml` and then close it.
5. Start the Oracle Sensor Edge Server using the the `userThreads` option. For example: `java -jar oc4j.jar -userThreads`

1.4.1.2.2 Creating the JMS Dispatcher To create a JMS dispatcher that enables the Oracle Sensor Edge Server to send event messages to the JMS topic queue, `edgeTopic`:

1. Within `<CurrentDispatcher>`, change the element content of `<DispatcherName>` to `JMS Dispatcher`.
2. Compare the `<ParameterInst>` elements with the `<Parameter>` attributes for the following:
   - `provider`

     The URI of the OC4J instance where the `edgeTopic` queue is stored. This URI is used internally by OC4J ORMI to connect to the server. For example, enter ormi://oc4j.us.oracle.com.
   - `username`

     The user name of the administrator of the OC4J instance where the `edgeTopic` queue is stored.
   - `password`

     The password used by the administrator of the OC4J instance where the `edgeTopic` queue is stored.
   - `ack`

     Set the acknowledgement mode to `CLIENT_ACKNOWLEDGE or AUTO_ACKNOWLEDGE`. The default mode is `AUTO_ACKNOWLEDGE`.
3. Save `edgeserver.xml` and restart the Oracle Sensor Edge Server. The server then loads the JMS Dispatcher upon startup.

1.4.1.3 Configuring the Dispatcher to Send Event Messages to a Web Service

A client device or application can register a SOAP call which the Oracle Sensor Edge Server invokes when a new message must be delivered.

To configure the Web Service dispatcher to distribute event messages through Web Services

1. Within `<CurrentDispatcher>`, change the element content of `<DispatcherName>` to `WebService Dispatcher`.
2. Compare the `url` attribute defined in `<Parameter>` elements for the WebService dispatcher with that defined in the `<ParameterInst>` element for the `<CurrentDispatcher>`. The `url` attribute must point to the EndPoint (port) of the Web Service. For example, enter `http://localhost:8888/wsdl/mytest.wsdl`.

   The WSDL (Web Service Definition Language) document must contain the `portType of EdgeClientCallback` and the call, `processEvent`, as its child element. For more information about the message parameters of the WSDL and a

sample of the WSDL document used for generating the Web Service stub, refer to the *Oracle Application Server Wireless Developer's Guide*.

Save `edgeserver.xml` and restart the Oracle Sensor Edge Server. The Oracle Sensor Edge Server then loads the WebService dispatcher upon startup.

1.4.1.4 Configuring the Dispatcher to Send Event Messages Through HTTP

Configuring the dispatcher to route events to clients using HTTP 1.0 results in the Oracle Sensor Edge Server posting each event message to the client separately. Because the Oracle Sensor Edge Server performs these posts sequentially, if one post is blocked, then all following posts are also blocked.

To set the HTTP dispatcher:

1. Within `<CurrentDispatcher>`, change the element content of `<DispatcherName>` to `HTTP Dispatcher`.
2. Compare the `url` attribute defined in `<Parameter>` elements for the HTTP Dispatcher with that defined in the `<ParameterInst>` element for the `<CurrentDispatcher>`. The value for this attribute is the URL of the servlet, JSP, or CGI to which the Oracle Sensor Edge Server posts event messages whenever they are dispatched. To configure this dispatcher, enter the URL in the following format:

   ```
   http://hostname:port/serverPath
   ```

   If the Oracle Sensor Edge Server uses the HTTP dispatcher, then the client interface must tell the Oracle Sensor Edge Server how (and when) to call it.
3. Save `edgeserver.xml` and restart the Oracle Sensor Edge Server. The server then loads the HTTP Dispatcher upon startup. For more information on the configuring the dispatcher to route events through HTTP, refer to the tutorial, *Writing Sensor Based Applications Using JSP, on the Oracle Technology Network* (`http:/www.oracle.com/technology/`).

   Refer to the *Oracle Application Server Wireless Developer's Guide* for a description of the parameters posted by the Oracle Sensor Edge Server to the client application.

1.4.1.5 Using the Nulldispatcher

The Nulldispatcher discards all of the events passed to it. These events are not saved or spooled. Use this dispatcher only if you want to prevent the Oracle Sensor Edge Server from dispatching events.

To set the Nulldispatcher:

1. Within `<CurrentDispatcher>`, change the element content of `<DispatcherName>` to `Nulldispatcher`.
2. Save `edgeserver.xml` and restart the Oracle Sensor Edge Server. The Oracle Sensor Edge Server then loads the HTTP dispatcher upon startup.

1.5 Connecting Readers and Sensors

To enable the Oracle Edge Sensor Server to use a driver for a reader or sensor, you must edit the `<DeviceGroups>` section of `edgeserver.xml` to create a device, an instance of the driver.

1.5.1 Creating a Device

To create a device:

1. Locate the <DeviceGroup> to which to add the device. If needed, add a new <DeviceGroup> element. For more information on the children of the <DeviceGroup>, refer to Section 1.2.4.
2. Add a new <Device> section. (You can copy a <Device> section from an existing device to use as a template.)
3. Set the name of the device within the <Name> tag.
4. Set the element definition in the <DriverName> tag to the name of a driver listed in the <DriverList> section.
5. Within the <Extension> element, set the value of the reference attribute to that of the driver.
6. Set the <ParameterInsts> to match the attribute values set for the driver's <Parameters> element.
7. If needed, add a filter instance to this device by defining the <FilterInsts> section.
8. Save edgeserver.xml and restart the Oracle Sensor Edge Server to instantiate the device.

Example 1–7 illustrates configuring the Oracle Edge Sensor Server to use a device called stack 1, an instance of a driver called Edge Device Driver.

Example 1–7 Configuring a Device

```
<DeviceGroups>
    <DeviceGroup>
      <DeviceList>
        <Device>
          <Name>stack1</Name>
          <DriverName>Edge Device Driver</DriverName>
          <Extension reference="49"/>
          <ParameterInsts>
            <ParameterInst>
              <Name>PortNo</Name>
              <ParameterMetaData reference="51"/>
              <Value>7878</Value>
            </ParameterInst>
            <ParameterInst>
              <Name>IPAddress</Name>
              <ParameterMetaData reference="52"/>
              <Value>152.69.156.193</Value>
            </ParameterInst>
          </ParameterInsts>
          <FilterInsts/>
        </Device>
   </DeviceList>
      <EventCollectWaitTime>500</EventCollectWaitTime>
      <FilterInsts id="108"/>
      <IsDefault>true</IsDefault>
      <IsSystem>true</IsSystem>
      <Name>Unassigned</Name>
    </DeviceGroup>
  </DeviceGroups>
```

1.6 Enabling Devices to Filter Events

A filter instance is an instantiated object of a filter. Whenever a filter is applied to a device (or to a device group), a filter instance is created, enabling the device or device group to use the filter. Filter instances can be attached to either a specific device or to a device group. Some filters are written as group-level filters and can only be attached to a device group. Likewise, some filters are written only for device-level filtering and only function if they are attached to a specific device. Fore more information on creating filters and configuring `edgeserver.xml` to filter events at both the device group and device level, refer to Section 3.3.

1.6.1 Creating a Filter Instance

To create a filter instance:

1. Within the `<DeviceGroups>`, locate the `<Device>` to which to add the filter instance.
2. Add a `<FilterInst>` element to the device's `<FilterInsts>` section.
3. For the `<Extension>` element, specify the name of the filter for the `class` attribute, and the `id` of the filter for the `reference` attribute.
4. Set the `<ParameterInsts>` to match the attribute values set for the filter's `<Parameters>` element.
5. Save `edgeserver.xml` and then restart the Oracle Sensor Edge Server.

   Example 1–8 illustrates configuring a device to use a filter instance by defining the `<FilterInsts>` element. In this example, the PalletPassFilter has a `<Sequence>` value set a 1. The number specified within the `<Sequence>` tag determines the order in which filter instances are applied. For more information or setting the order of the filter instances, refer to Section 3.3.1.1.

Example 1–8 Configuring a Filter Instance for a Device

```
<FilterInsts>
      <FilterInst>
          <Sequence>1</Sequence>
          <FilterName>PalletPassFilter</FilterName>
          <Extension class="Filter" reference="50"/>
          <ParameterInsts>
            <ParameterInst>
              <ParameterMetaData reference="52"/>
              <Value>11</Value>
              <Name>ExitEventThresholdTime</Name>
            </ParameterInst>
            <ParameterInst>
              <ParameterMetaData reference="53"/>
              <Value>12</Value>
              <Name>EventCollectControlTime</Name>
            </ParameterInst>
          </ParameterInsts>
          <NeedReload>false</NeedReload>
          <Name>PalletPassFilter</Name>
        </FilterInst>
 </FilterInsts>
```

2

Managing Drivers

This chapter, through the following sections, describes how to manage and configure the drivers for the Oracle Sensor Edge Server.

- Section 2.1, "Managing Drivers"
- Section 2.2, "Configuring the Pre-Seeded Drivers"

2.1 Managing Drivers

Drivers read events. Because a driver object is not executed, the Oracle Sensor Edge Server can only use a driver's functionality if you create an instance of a driver called a device. For more information on creating devices, refer to Section 1.5.1.

The `<DriverList>` element of `edgesever.xml` (illustrated in Example 2–1) lists all of the driver that have been uploaded to (and can be used by) the Oracle Sensor Edge Server.

Example 2–1 The <DriverList> Element of edgeserver.xml

```
<DriverList id="23">
    <Driver id="24">
      <ClassName>oracle.edge.impl.driver.EdgeSimulator</ClassName>
      <Description>This is internal simulator</Description>
      <Name>Edge Simulator Driver</Name>
      <Parameters id="25">
        <Parameter id="26" name="FileName"
         defaultValue="..\..\edge\config\Simulation.xml"
         description="Simulator's configuration file" encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
    <Driver id="27">
      <ClassName>oracle.edge.impl.driver.AlienReader</ClassName>
      <Description>This is an alien device</Description>
      <Name>AlienDevice</Name>
      <Parameters id="28">
        <Parameter id="29" name="PortNo" defaultValue="23"
                           description="Alien reader's open port number that edge
                           device listens to"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="30" name="IPAddress" defaultValue="144.25.171.23"
```

```
                    description="Alien reader's IP address"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="31" name="UserName" defaultValue="alien"
                    description="Alien reader's access user"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="32" name="Password" defaultValue="password"
                    description="Alien reader's access password"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="33" name="AntennaSeqIdList" defaultValue=""
                    description="List of identifiers to identify each
                    antenna" encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="34" name="AntennaMappedDeviceNameList" defaultValue=""
                    description="List of mapped device names associated
                    with each antenna"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Driver>
<Driver id="35">
  <ClassName>oracle.edge.impl.driver.IntermecReader</ClassName>
  <Description>This is Intermec reader: IntelliTag 500</Description>
  <Name>IntermecDevice</Name>
  <Parameters id="36">

    <Parameter id="37" name="PortNo" defaultValue="6543"
                    description="Reader's open port number that edge device
                    listens to"
                    encrypted="false">
      <valueType type="int"/>
    </Parameter>

    <Parameter id="38" name="IPAddress" defaultValue="192.168.0.52"
                    description="Reader's IP address" encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="39" name="AntennaSeqIdList" defaultValue=""
                    description="List of identifiers to identify each
                    antenna" encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="40" name="AntennaMappedDeviceNameList" defaultValue=""
                    description="List of mapped device names associated
                    with each antenna"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Driver>
<Driver id="41">
```

```
      <ClassName>oracle.edge.impl.driver.EdgeEventDevice</ClassName>
      <Description>This is EMS reader.</Description>
      <Name>EMSDevice</Name>
      <Parameters id="42">
        <Parameter id="43" name="PortNo" defaultValue="6666"
                           description="Reader's open port number that edge device
                           listens to"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="44" name="IPAddress" defaultValue="144.25.168.131"
                           description="Reader's IP address" encrypted="false"
<valueType type="string"/>
        </Parameter>
        <Parameter id="45" name="AntennaSeqIdList" defaultValue=""
                           description="List of identifiers to identify each
antenna" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="46" name="AntennaMappedDeviceNameList" defaultValue=""
                           description="List of mapped device names associated
                           with each antenna"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
    <Driver id="47">
      <ClassName>oracle.edge.impl.driver.EdgeDevice</ClassName>
      <Description>Edge Device Driver</Description>
      <Name>Edge Device Driver</Name>
      <Parameters id="48">
        <Parameter id="49" name="PortNo" defaultValue="23"
                           description="Edge device's open port number that edge
                           device listens to"
                           displayName="Port Number" encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="50" name="IPAddress" defaultValue="" description="Edge
                           device's IP address"
                           displayName="IP Address" encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
```

2.2 Configuring the Pre-Seeded Drivers

The Oracle Sensor Edge Server provides the following drivers out of the box:

- EdgeSimulator (Section 2.2.1)
  - Version: 10.1.2
- Alien (Section 2.2.2)
  - Model: 9RE-001
  - SDK Version: Alien SDK Version 2.1.0

- Intermec (Section 2.2.3)
  - Models: Penn Reader, Delaware Reader, PCMCIA Reader
  - SDK Version: COM API Version 2.0
- Patlite (Section 2.2.4)
  - Model: New PHE-3FB PC-Controlled Light
  - Protocol: PHE-3FB System Control Protocol

2.2.1 Configuring the EdgeSimulator

The EdgeSimulator (the Edge Simulator Driver) generates events to simulate a real device. In general, you use the EdgeSimulator to test configurations and deployment designs; however, you can also use it for internal functional testing to see how events are processed throughout the system. The EdgeSimulator acts the same as any driver, except that instead of connecting to a physical device to read events, it takes parameters from an input file (such as Example 2–5) as instructions on when to generate fake events. This begins as soon as the device starts (which, in turn, starts when the Oracle Sensor Edge Server starts).

2.2.1.1 Defining the Parameters of the EdgeSimulator

To configure the EdgeSimulator, define the `<Parameter>` tag of the `<Driver>` element by entering the name of the input file, which is an XML file describing the configuration for the EdgeSimulator. In Example 2–2, the `defaultValue` attribute for this file in the `<Parameter>` tag is `..\..\edge\config\Simulation.xml`.

The file resides at:

`ORACLE_HOME/edge/config`

Example 2–2 The Driver Definition for the EdgeSimulator in edgeserver.xml

```
<DriverList id="23">
    <Driver id="24">
      <ClassName>oracle.edge.impl.driver.EdgeSimulator</ClassName>
      <Description>This is the internal simulator</Description>
      <Name>Edge Simulator Driver</Name>
      <Parameters id="25">
        <Parameter id="26" name="FileName"
         defaultValue="..\..\edge\config\Simulation.xml" description="Simulator's
         configuration file" encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
```

The input file tells the simulator how to generate the fake events using the following instructions:

- `<EventList>`

  The `<EventList>` element defines a loop. This element is also the main block that groups all of the other instructions together. `<EventList>` has one attribute, `repeat`, which must be present to control looping. The value for `repeat` must be a decimal number from 0 to `LONG_MAX`. To generate events only once, set the

repeat attribute to 1. Setting repeat to n results in all instructions looping n times. Setting repeat to 0 disables the block and causes the parser to skip it.

Example 2–3 illustrates the syntax for generating two events, pausing, generating two more events, and then looping 20 times:

Example 2–3 Defining a Loop

```
<EventList repeat='20'>
<Event> … </Event>
<Event> … </Event>
<EventInterval>…</ EventInterval>
<Event> … </Event>
<Event> … </Event>
</EventList>
```

You can include any number of instructions inside the <EventList> element. The order in which they are defined is the order in which they are executed.

- <EventInterval>

  The EventInterval element instructs the Simulator to pause for a certain period of time before proceeding. This is usually used to throttle the data rate. A decimal number defines the time period, in milliseconds, to wait for before executing the next instruction. Example 2–4 illustrates instructions for the Simulator to wait for half a second between each event and three seconds between loops:

Example 2–4 EventInterval

```
<EventList repeat='20'>
   <Event> … </Event>
   <EventInterval>500</ EventInterval>
   <Event> … </Event>
   <EventInterval>500</ EventInterval>
   <Event> … </Event>
   <EventInterval>3000</ EventInterval>
</EventList>
```

- <Event>

  The <Event> element tells the Simulator to send an event. The child elements (described in Table 2–1) control the event's fields.

Table 2–1 Event Elements for the Simulator

Event Field	Value
<type>	The number value that corresponds to the type of event. Table 1–1 describes the values of the <type> field. The *Oracle Application Server Wireless Developer's Guide* describes the values for the <type> field in further detail.
<subtype>	The number value for the subtype. For example, the subtype value in Example 2–5 corresponds with a General Instruction Event, which is an event sent by application or a device to tell a specific device to perform an operation. In Example 2–5, the value of 1 turns on the device. Refer to the *Oracle Application Server Wireless Developer's Guide* for more information on Instruction Events.

Table 2–1 (Cont.) Event Elements for the Simulator

Event Field	Value
<id>	The text value of this field identifies a tag (that is, a read or target) to an event instruction. In Example 2–5, one of the <id> values for a tag is *03ffff045679*.
<data>	The tag data. This is an optional field
<deviceName>	The name of the device or application that generates the event. The <deviceName> enables the Simulator to appear as if it is another device when generating events.

Example 2–5 illustrates an input file which includes two groups of events: the first one runs only once and the second runs 20 times.

Example 2–5 Simulator Input File

```
<EdgeEventSimulation>
      <EventList repeat='1'>
         <Event>
            <type>100</type>
            <subtype>1</subtype>
            <id>03ffff045679</id>
            <data>No Data</data>
            <deviceName>My Device</deviceName>
        </Event>
   <EventInterval>500</ EventInterval>
      <Event>
         <type>100</type>
            <subtype>1</subtype>
            <id>03ffff045680</id>
            <data>No Data</data>
            <deviceName>My Device</deviceName>
     </Event>
          <EventInterval>3000</ EventInterval>
   </EventList>
   <EventList repeat='20'>
      <Event>
         <type>100</type>
         <subtype>1</subtype>
         <id>04ffff045679</id>
         <data>No Data</data>
         <deviceName>My Device</deviceName>
    </Event>
         <EventInterval>500</ EventInterval>
     <Event>
         <type>100</type>
          <subtype>1</subtype>
          <id>04ffff045680</id>
          <data>No Data</data>
          <deviceName>My Device</deviceName>
     </Event>
   </EventList>
</EdgeEventSimulation>
```

Although the format of the Event type is fixed, you can extend the Event type by mapping its fields to different meanings depending on the type of event.

2.2.1.2 Connecting the Simulator to the Oracle Sensor Edge Server

To connect the Simulator to the Oracle Sensor Edge Server, create a device from the driver by defining the <Name>.

To create a device for the Simulator driver:

1. Locate the <DeviceGroup> to which to add the Simulator device. If needed, add a new <DeviceGroup> element. For more information on the children of the <DeviceGroup>, refer to Section 1.2.4.
2. Add a new <Device> section for the Simulator. (You can copy a <Device> section from an existing device to use as a template.
3. Set the name of the device within the <Name> tag.
4. Set the element definition in the <DriverName> tag to Edge Simulator Driver.
5. Within the <Extension> element, set the value of the reference attribute to that of the Edge Simulator Driver.
6. Within the <ParameterInsts> element, set the location of the simulator's input XML file.
7. Save edgeserver.xml and restart the Oracle Sensor Edge Server to instantiate the device.

2.2.2 Configuring the AlienDevice Driver

The AlienDevice driver, which is defined within the <DriverList> element of edgeserver.xml (Example 2–6) supports all of the Alien Technology RFID readers. The ALR series of readers has been tested for this release of the Oracle Sensor Edge Server. For more information on the ALR series, refer to:

http://www.alientechnology.com/

Configuring the AlienDevice driver includes:

- Finding the IP Address of the Alien RFID reader using the discoverer included in the RFID Gateway demo software provided by Alien Technology (Section 2.2.2.1).
- Connecting the Alien RFID reader to a Web browser (Section 2.2.2.2).
- Connecting the Alien RFID reader to the Oracle Edge Server by creating a device using the AlienDevice driver (Section 2.2.2.3).

Example 2–6 The Alien Device Driver

```
<Driver id="27">
      <ClassName>oracle.edge.impl.driver.AlienReader</ClassName>
      <Description>This is an alien device</Description>
      <Name>AlienDevice</Name>
      <Parameters id="28">
        <Parameter id="29" name="PortNo" defaultValue="23"
                           description="Alien reader's open port number that edge
                           device listens to"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="30" name="IPAddress" defaultValue="127.0.0.1"
```

```
                    description="Alien reader's IP address"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="31" name="UserName" defaultValue="alien" description="Alien
                    reader's access user"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="32" name="Password" defaultValue="password"
                    description="Alien reader's access password"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="33" name="AntennaSeqIdList" defaultValue=""
                    description="List of identifiers to identify each
                    antenna" encrypted="false">
      <valueType type="string"/>
    </Parameter>
    <Parameter id="34" name="AntennaMappedDeviceNameList" defaultValue=""
                    description="List of mapped device names associated
                    with each antenna"
                    encrypted="false">
      <valueType type="string"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Driver>
```

2.2.2.1 Finding the IP Address of the Alien RFID Reader

By default, the Alien RFID reader uses DHCP (Dynamic Host Configuration Protocol) to get its IP address upon connection to the network. To discover the IP address without a DCHP server, use the RFID Gateway Demo software as follows:

1. Install the RFID Gateway Demo software.
    1. Run the `setup.exe` (on the Alien Technology CD).
    2. Select **Install RFID Gateway Demo Software** and follow the installation instructions. A folder called *Alien RFID Gateway* appears in the *Programs* folder of the *Start* menu.
2. When the RFID Gateway Demo Software is installed, use it to discover the IP address as follows:
    1. Start the Alien RFID software by selecting `Alien RFID Gateway` (located in the *Alien RFID Gateway* folder of the *Start* menu). Upon startup, the Alien RFID Gateway software scans the serial ports of the computer and the network and displays a list of the Alien RFID readers and their IP addresses in the window that appears. For example, a reader might be listed as *Alien RFID Reader@144.25.171.209*.
    2. Get the IP address of the Alien RFID Reader from the list. Using this address, you can configure the Alien RFID reader to the Web browser.

2.2.2.2 Connecting the Alien RFID Reader to a Web Browser

Because the Alien reader has a built-in Web Server, you can connect it to a Web browser by pointing the browser to the IP address of the Alien device, such as `http://144.25.171.209`. When prompted, enter a user name and password.

Because the default settings should be correct, you can then create a device from the RFID reader, which is an instance of the device. For more information on creating devices, see Section 1.5.

2.2.2.3 Creating a Device for the Alien RFID Reader

To connect the Alien RFID reader to the Oracle Sensor Edge Server, you must create a device (that is, an instance of) the Alien RFID reader.

To create a device for the Alien RFID reader:

1. Locate the `<DeviceGroup>` to which to add the Alien RFID reader's device. If needed, add a new `<DeviceGroup>` element. For more information on the children of the `<DeviceGroup>`, refer to Section 1.2.4.
2. Add a new `<Device>` section for the Alien Device. (You can copy a `<Device>` section from an existing device to use as a template.
3. Set the name of the device within the `<Name>` tag.
4. Enter `AlienDevice` as the element content for the `<DriverName>` tag.
5. Within the `<Extension>` element, set the value of the `reference` attribute to that of the AlienDevice driver.
6. Within the `<ParameterInsts>` element, create the following `<ParameterInst>` elements as described in Section 1.2.4.1. Table 2–2 describes the element content for the `<ParameterInst>` elements for an Alien Reader device.

Table 2–2 <ParameterInst> Elements for the Alien Reader Device

<Name>	<ParameterMetaData>	<Value>
`IP Address`	Enter the reference number for the extension reference that points to `<Parameter>` tag containing the IPAddress attribute.	Enter the hostname or IP address of the machine running the Device Controller as its element content. If the machine runs on the same machine as the Oracle Sensor Edge Server, enter 127.0.0.1
`PortNo`	Enter the reference number for the extension reference that points to `<Parameter>` tag containing the `Portno` attribute.	Enter the port number used to communicate with the device (23 is the default).
`Username`	Enter the reference number for the extension reference that points to `<Parameter>` tag containing the `Username` attribute.	Enter the access user of the Alien RFID reader.

Table 2–2 (Cont.) <ParameterInst> Elements for the Alien Reader Device

<Name>	<ParameterMetaData>	<Value>
Password	Enter the reference number for the extension reference that points to <Parameter> tag containing the password attribute.	Enter the password for the Alien RFID reader
AntennaSeqIDList	Enter the reference number for the extension reference that points to <Parameter> tag containing the AntennaSeqIDList attribute	Enter a list of identifiers to identify each antenna.
AntennaMappedDeviceNameList	Enter the reference number for the extension reference that points to <Parameter> tag containing the AntennaMappedDeviceNameList attribute	Enter a list of mapped device names associated with each antenna.

7. Save edgeserver.xml and restart the Oracle Sensor Edge Server to instantiate the device.

2.2.3 Configuring the IntermecDevice Driver

The IntermecDevice driver, which is defined within the <DriverList> element of edgeserver.xml (Example 2–7), supports all of the RFID readers manufactured by Intermec, including the OEM Reader (Microwave, UHF), the PC Reader (PCMCIA), and the Fixed Reader (Serial or Ethernet). Other Intermec readers that support the Intellitag IDK also work with the IntermecDevice driver. For more information, refer to

http://www.intermec.com

Example 2–7 The Driver Definition for the IntermecDevice Driver in edgeserver.xml

```
</Driver>
    <Driver id="35">
      <ClassName>oracle.edge.impl.driver.IntermecReader</ClassName>
      <Description>This is Intermec reader: IntelliTag 500</Description>
      <Name>IntermecDevice</Name>
      <Parameters id="36">
         <Parameter id="37" name="PortNo" defaultValue="6666"
                            description="Reader's open port number that edge
                            device listens to"
                            encrypted="false">
          <valueType type="int"/>
        </Parameter>
          <Parameter id="38" name="IPAddress" defaultValue="127.0.0.1"
                             description="Reader's IP address"
                             encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="39" name="AntennaSeqIdList" defaultValue=""
                           description="List of identifiers to identify each
                           antenna" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="40" name="AntennaMappedDeviceNameList" defaultValue=""
                           description="List of mapped device names associated
                           with each antenna"
                           encrypted="false">
```

```
      <valueType type="string"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Driver>
```

Requirements

The IntermecDevice requires the following components, which are bundled and shipped with the Intermec driver:

- IntelliTag IDK

  The IntelliTag IDK (the IDK) is a set of Intermec-supported software libraries and tools. This library, which is the only supported method of communicating with Intermec devices, is supported only on the Windows 32 platform (that is, Windows 2000 and Windows XP). The IntelligTag IDK is available at the Oracle Technology Network (`http://www.oracle.com/technology/`)

- Oracle Sensor Edge Server Device Controller

  The Oracle Sensor Edge Server Device Controller (the Device Controller) communicates with the local IDK API and exposes a network protocol that enables the Oracle Sensor Edge Server to communicate with the IDK.

- IntermecDevice Driver

  The IntermecDevice driver is the counterpart to the Device Controller, as it communicates with the Device Controller to drive the underlying devices.

The Oracle Sensor Edge Server can run on the same server as the Device Controller and the IDK, or on a separate server. Because the Intermec hardware exposes a Windows 32- based API, you must run the Oracle Sensor Edge Server on a Windows box or dedicate another Windows machine to only the Device Controller and the IDK.

Configuration Steps

Configuring the IntermecDevice driver involves the following tasks:

- Installing the IDK (Section 2.2.3.1).
- Registering Serial and PCMCIA readers (Section 2.2.3.2).
- Configuring the Intermec readers (Section 2.2.3.3).
- Using the Intermec tools to test the Intermec readers (Section 2.2.3.4).
- Installing the Device Controller (Section 2.2.3.5).
- Starting the Device Controller for the Intermec reader (Section 2.2.3.6).
- Creating a device using the IntermecDevice driver to enable communication between the Oracle Sensor Edge Server and the Device Controller for the Intermec reader (Section 2.2.3.7).

2.2.3.1 Installing the IntelliTag IDK

To install the IDK and tools:

1. Connect the RFID reader to either a serial port, the PCMCIA slot of a PC, or a network segment. Connect a serial reader with a null modem (2/3 swap) cable. For more information, refer to the Intermec documentation.

2. Extract the content of the compressed file to a temporary directory, such as `(c:\temp):ORACLE_HOME\edge\lib\IDK_Beta_4.0.1.tgz`. (You can also download the latest version of the IDK from Intermec's Web site)
3. Install the IDK by running the install at `C:\temp\setup.exe`.
4. At the installer's first page, select **Next**.
5. In the *Agreements* page, read the license agreement and select **Yes** or **No**. Selecting **No** prevents you from proceeding.
6. On the next page, enter your name, the company name, and then select **Anyone who uses this computer**. Click **Next**.
7. Select a directory for the IDK. Use the default, if possible.
8. Select **Typical** for the setup type and then click **Next**.
9. Click **Next** to install the IDK and programs to the PC.

2.2.3.2 Registering the Serial and PCMCIA Readers

You must register devices for the Serial and PCMCIA readers. The following steps to register these devices to do not apply to Ethernet readers.

To register devices:

1. From the *Start* menu, select **Intellitag IDK** and then **Device Registry Application**.
2. Select the *Register Readers* tab. Be sure that the reader is connected.
3. Select an existing reader from the *Select Reader* drop-down list or enter the name of a new reader in the *New Reader Name* field and then click **Register New Reader**.
4. In the *Port Name* field, enter the name of the port that you use to connect to the reader.

Accept the default settings (unless you have changed the device).

2.2.3.3 Configuring Readers

Once you register a reader, you next configure it by editing the `rfconfig.ini` file. Open the `rfconfig.ini` file from the *Start* menu, select **IntelliTag IDK** and then **rfconfig.ini**. The file, which opens in Notepad is formatted as a standard Windows INI file. Each section of the file represents a new reader configuration, as illustrated by the `[Reader_One]` section in Example 2–8.

Example 2–8 Configuring rfconfig.ini

```
[Reader_One]
RFID_SWTT_FILE_NAME=C:\Program Files\Intermec\Intellitag IDK\swtt.ini
RFID_ATTR_TYPE=IT500 UAP Reader
IT500_PORT_TYPE=TCPIP
IT500_PORT_NUMBER=6543
IT500_CONNECT_TRIES=1
IT500_PORT_NAME=192.168.200.47
IT500_DEBUG_FILE_NAME="c:\IT500_Reader.log"
IT500_ANTENNA_TRIES=5
IT500_ANTENNAS=1 2 0 0 0 0 0 0
IT500_READ_TRIES=5
IT500_WRITE_TRIES=5
IT500_INTERR_DEBUG=0
IT500_READER_DEBUG=0
dll_name=C:\Program Files\Intermec\Intellitag IDK\it500.dll
```

```
IT500_IDENTIFY_TRIES=1
IT500_INITIALIZATION_TRIES=1
IT500_SIM_TAGS=5
IT500_IDENTIFY_READ_END_ADDR=17
IT500_HARDWARE_TYPE_CHECK=0
IT500_AUTOID_TIMEOUT=20
```

Although you can rename `[Reader One]` to any name, note this name for future reference. Modify (or verify) the following settings for the `rfconfig.ini` file:

- `IT500_PORT_TYPE`

  This parameter tells the API the type of connection to use, such as TCPIP for a network reader or serial or PCMCIA reader.

- `IT500_PORT_NAME`

  If it is a serial or PCMCIA reader, this parameter sets the name of the reader that you registered (see Section 2.2.3.2). For network readers, this is the hostname or IP address of the reader.

- `IT500_PORT_NUMBER`

  This parameter specifies the TCP / IP port used to connect to the reader. The default setting is 6543. This parameter should only be defined for a network reader.

- `IT500_ANTENNAS`

  This is a mask for the antennae that are active and connected to the reader. The first digit corresponds to the first antenna. For example, if you have Antennas 1 and 3 connected to the reader and Antenna 1 is the first antenna, then set the parameter to `IT500_ANTENNAS=1 0 3 0 0 0 0 0`. For four antennae connected consecutively, set this parameter to `T500_ANTENNAS=1 2 3 4 0 0 0 0`.

  Save the Notepad file and then close it after you complete the configuration.

2.2.3.4 Testing the Readers

To test the reader using Intermec tools:

1. From the *Programs* folder of the *Start* menu, click **Intellitag IDK** and then **RF Tag Map**.
2. Click **Select** to select a reader configuration. A dialog box appears listing the configurations defined in the `rfconfig.ini` file.
3. Select a reader configuration and then click **Select**. The **Open** button is activated.
4. Click the **Open** button to connect to the device
5. If then reader is connected properly, then the buttons in the *Tag Map* section are enabled.
6. Click **Start** to start the reader.
7. Wave the sample tags in front of the antenna. The tag ID and payload should be read and appear on the screen.

2.2.3.5 Installing the Oracle Sensor Edge Server Device Controller

Install the Device Controller by extracting the `ORACLE_HOME\edge\controller\deviceController.zip` file into the `C:\` directory. This extracts the Device Controller files into the `C:\controller` directory.

2.2.3.6 Starting the Oracle Sensor Edge Server Device Controller

To start the device controller:

1. Start a command-line console (`cmd.exe`)
2. Navigate to the `C:\deviceController` directory and then run the following command:

   ```
   startIntermec.bat <ReaderName> <Port>
   ```

   where `<ReaderName>` is the is the configuration name of the reader in the rfconfig.ini file and `<Port>` is the port on which the IntermecDevice driver listens so that it can communicate with this Device Controller.

   For example, to start the Device Controller for the reader called Penn_A at port 6666, run the following command:

   ```
   startIntermec.bat Penn_A 6666
   ```

   After the Device Controller for the reader starts, create a device from the IntermecDevice driver that enables the Oracle Sensor Edge Server to communicate to the reader through this Device Controller.

2.2.3.7 Configuring the Oracle Sensor Edge Server to Communication with the Device

You must create a device, an instance of the IntermecDriver driver, to enable the Oracle Sensor Edge Server to communicate with the Device Controller by editing the `<DeviceGroups>` element of `edgeserver.xml` (Example 2–9).

Example 2–9 The Device Configuration for the IntermecDevice Driver

```
<DeviceGroups id="81">
    <DeviceGroup id="82">
      <DeviceList id="83">
         <Device id="84">
          <Name>Intermec Device</Name>
          <DriverName>IntermecDevice</DriverName>
       <Extension reference="35"/>
          <ParameterInsts id="85">
            <ParameterInst id="86">
              <Name>PortNo</Name>
              <ParameterMetaData reference="37"/>
              <Value>6666</Value>
            </ParameterInst>
            <ParameterInst id="87">
              <Name>IPAddress</Name>
              <ParameterMetaData reference="38"/>
              <Value>192.168.0.52</Value>
            </ParameterInst>
            <ParameterInst id="88">
              <Name>AntennaSeqIdList</Name>
              <ParameterMetaData reference="39"/>
              <Value>12000000</Value>
            </ParameterInst>
            <ParameterInst id="89">
```

```
            <Name>AntennaMappedDeviceNameList</Name>
           <ParameterMetaData reference="40"/>
            <Value>IT500_READER</Value>
          </ParameterInst>
        </ParameterInsts>
        <FilterInsts id="90"/>
      </Device>
    </DeviceList>
    <EventCollectWaitTime>500</EventCollectWaitTime>
    <FilterInsts id="91"/>
    <IsDefault>false</IsDefault>
    <IsSystem>true</IsSystem>
    <Name>Unassigned</Name>
  </DeviceGroup>
</DeviceGroups>
```

To create a device:

1. Locate the <DeviceGroup> to which to add the device for the IntermecDevice reader. If needed, add a new <DeviceGroup> element. For more information on the children of the <DeviceGroup>, refer to .
2. Add a new <Device> section for the Intermec Device. (You can copy a <Device> section from an existing device to use as a template.
3. Set the name of the device within the <Name> tag.
4. Enter IntermecDevice as the element content for the <DriverName> tag.
5. Within the <Extension> element, set the value of the reference attribute to that of the IntermecDevice driver.
6. Within the <ParameterInsts> element, create the following <ParameterInst> elements as described in Section 1.2.4.1. Table 2–3 describes the element content for the <ParameterInst> elements for the device.

Table 2–3 <ParameterInst> Elements for the Intermec Device Driver

<Name>	<ParameterMetaData>	<Value>
IP Address	Enter the reference number for the extension reference that points to <Parameter> tag containing the IPAddress attribute	Enter the hostname or IP address of the machine running the Device Controller as its element content. If the machine runs on the same machine as the Oracle Sensor Edge Server, enter 127.0.0.1.

Table 2–3 (Cont.) <ParameterInst> Elements for the Intermec Device Driver

<Name>	<ParameterMetaData>	<Value>
PortNo	Enter the reference number for the extension reference that points to <Parameter> tag containing the Portno attribute.	Enter the port number used to start the Device Controller (6666 is the default).
AntennaSeqIDList	Enter the reference number for the extension reference that points to <Parameter> tag containing the AntennaSeqIDList attribute.	Enter a list of identifiers to identify each antenna.
AntennaMappedDeviceNameList	Enter the reference number for the extension reference that points to <Parameter> tag containing the AntennaMappedDeviceNameList attribute.	Enter a list of mapped device names associated with each antenna.

7. Save `edgeserver.xml` and restart the Oracle Sensor Edge Server to instantiate the device. For more information on starting and stopping the Oracle Sensor Edge Server, refer to Section 1.3.

2.2.4 Configuring the Patlite Driver

Unlike the RFID readers or other sensors, the Patlite series of lightstacks and trees do not generate events, but instead act as indicator lights and signals. Sending events to Patlite lightstacks and trees turns on lights or causes them to blink for certain intervals.

Configuring the Patlite driver involves the following tasks:

- Installing the Patlite hardware (Section 2.2.4.1).
- Configuring the Device Controller for the Patlite device (Section 2.2.4.2).
- Starting the Device Controller for the Patlite device (Section 2.2.4.3).
- Configuring the Oracle Sensor Edge Server to communicate with the Device Controller for the Patlite device by creating a device instance (Section 2.2.4.4).

Supported Patlite Devices

Patlite's products include those that support both Serial and Ethernet connection. The current version of the Patlite driver in this release supports the Serial connection only.

2.2.4.1 Installing the Patlite Hardware

To connect the Patlite device, you must have the following hardware:

- A free RS232C communication port
- A Female / Female, nine-pin RS232 cable with a straight-through pin type (such as a modem cable).

To set up the hardware:

1. Connect the lightstack to a power supply.
2. Connect one end of the serial cable to the lightstack.
3. Connect the other end of the serial cable to the serial port.

2.2.4.2 Configuring the Oracle Edge Server Device Controller for the Patlite Device

After you install the Device Controller (as described in Section 2.2.3.5), edit the `deviceController/config/dcconfig.xml` file as follows:

- Change the `comName` parameter to the com port that you are using.
- If the default port is currently in use on the local machine, change the value for `lcPort` to an available TCP/IP port number.

  In Example 2–10, the `dcconfig.xml` file uses `COM3` as the value for `comName` and the default port of `7878` for the `lcPort` parameter.

Example 2–10 Configuring the Com Port in dcconfig.xml

```
<?xml version="1.0"?>
   <Configuration>
      <ConfigParam name="lcPort" value="7878" />
      <ConfigParam name="comName" value="COM3" />
   </Configuration>
```

Note: The Intermec and Lightstack Device Controllers are available for download from Oracle Technology Network (`http://www.oracle.com/technology/`)

2.2.4.3 Starting the Device Controller for the Patlite Device

To start the device controller:

1. Navigate to `deviceController/deploy/win`.
2. Run `startLight.bat`. A message similar to Example 2–10 appears.

Example 2–11 Status Message

```
C:\deviceController\deploy\win>startlight
Local ip is: 144.25.168.146
Establishing the listener at port:  [7878]...
Waiting for connections...
```

After the Device Controller starts, you can enable communication between the Oracle Sensor Edge Server and the Device Controller for the Patlite device by creating a device from the Patlite Device driver.

2.2.4.4 Configuring the Oracle Sensor Edge Server to Communicate with the Device Controller

You must create a device, an instance of the Patlite Device driver, to enable the Oracle Sensor Edge Server to communicate with the Device Controller.

To create a device:

1. Locate the `<DeviceGroup>` to which to add the device for the Patlite driver. If needed, add a new `<DeviceGroup>` element. For more information on the children of the `<DeviceGroup>`, refer to Section 1.2.4.
2. Add a new `<Device>` section for the Patlite device. (You can copy a `<Device>` section from an existing device to use as a template.
3. Set the name of the device within the `<Name>` tag.
4. Enter `PatliteDriver` as the element content for the <DriverName> tag.

5. Within the <Extension> element, set the value of the reference attribute to that of the Patlite driver.
6. Within the <ParameterInsts> element, create the following <ParameterInst> elements as described in Section 1.2.4. Table 2–4 describes the element content for the <ParameterInst> elements for the device.

Table 2–4 <ParameterInst> Elements for the Patlite Device

<Name>	<ParameterMetaData>	<Value>
IP Address	Enter the reference number for the extension reference that points to <Parameter> tag containing the IPAddress attribute.	Enter the hostname or IP address of the machine running the Device Controller as its element content.
PortNo	Enter the reference number for the extension reference that points to <Parameter> tag containing the Portno attribute.	Enter the port number set in the dcconfig.ini file (7878 is the default)

7. Save edgeserver.xml and restart the Oracle Sensor Edge Server to instantiate the device. For more information on starting and stopping the Oracle Sensor Edge Server, refer to Section 1.3.

3

Managing Filters

This chapter, through the following sections, describes how to manage and configure the filters for the Oracle Sensor Edge Server.

- Section 3.1, "Managing Filters"
- Section 3.2, "Defining the Parameters of the Pre-Seeded Filters"
- Section 3.3, "Enabling Event Filtering for Devices or Device Groups"

3.1 Managing Filters

A filter is a class that strains out unwanted events or translates higher-level events from groups or events or specific conditions. An event is a message that is sent from either a sensor device or an application that signals that a state has changed. The Oracle Sensor Edge Server, which receives the data from these sensor devices or applications, normalizes the contents of these event messages by putting them in a common data format and then applies filters to strip them of extraneous information or unwanted events.

The `<FilterList>` element (Example 3–1) of `edgeserver.xml` defines the filters that are available to the Oracle Sensor Edge Server.

Example 3–1 The <FilterList> Element of edgeserver.xml

```
<FilterList id="51">
    <Filter id="52">
      <ClassName>oracle.edge.impl.filter.PassFilter</ClassName>
      <Description>Filter redundant in range tag ids from a single
      reader.</Description>
      <Name>PassRedundantFilter</Name>
      <Parameters id="53">
        <Parameter id="54" name="ExitEventThresholdTime" defaultValue="800"
                           description="Time elapsed in
                           milliseconds since a tag has been seen last time"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Filter>
    <Filter id="55">
      <ClassName>oracle.edge.impl.filter.PalletPassFilter</ClassName>
      <Description>Filter redundant in range tag ids from a single
reader.</Description>
      <Name>PalletPassFilter</Name>
      <Parameters id="56">
```

```
    <Parameter id="57" name="ExitEventThresholdTime" defaultValue="800"
                       description="Time elapsed in milliseconds since a tag
                       has been seen last time"
                       encrypted="false">
      <valueType type="int"/>
    </Parameter>
    <Parameter id="58" name="EventCollectControlTime" defaultValue="1500"
                       description="Time elapsed in milliseconds since a
                       new tag has been detected last time"
                       encrypted="false">
      <valueType type="int"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Filter>
<Filter id="59">
  <ClassName>oracle.edge.impl.filter.PalletShelfFilter</ClassName>
  <Description>Filter redundant in range tag ids from a single reader.
 </Description>
  <Name>PalletShelfFilter</Name>
  <Parameters id="60">
    <Parameter id="61" name="ExitEventThresholdTime" defaultValue="800"
                       description="Time elapsed in milliseconds since a tag
                       has been seen last time"
                       encrypted="false">
      <valueType type="int"/>
    </Parameter>
    <Parameter id="62" name="EventCollectControlTime" defaultValue="1500"
                       description="Time elapsed in milliseconds since a new
                       tag has been detected last time"
                       encrypted="false">
      <valueType type="int"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Filter>
<Filter id="63">
  <ClassName>oracle.edge.impl.filter.ShelfFilter</ClassName>
  <Description>Filter redundant in range tag ids from a single reader.
  </Description>
  <Name>ShelfRedundantFilter</Name>
  <Parameters id="64">
    <Parameter id="65" name="ExitEventThresholdTime" defaultValue="800"
                       description="Time elapsed in milliseconds since a tag
                       has been seen last time"
                       encrypted="false">
      <valueType type="int"/>
    </Parameter>
  </Parameters>
  <Version>1.0</Version>
</Filter>
<Filter id="66">
  <ClassName>oracle.edge.impl.filter.CrossReaderRedundantFilter</ClassName>
  <Description>Filter redundant tag ids from multiple readers.</Description>
  <Name>CrossReaderRedundantFilter</Name>
  <Parameters id="67"/>
  <Version>1.0</Version>
</Filter>
<Filter id="68">
  <ClassName>oracle.edge.impl.filter.CheckTagFilter</ClassName>
```

```
    <Description>Check Tag Filter</Description>
    <Name>Check Tag Filter</Name>
    <Parameters id="69">
      <Parameter id="70" name="CheckTagId" defaultValue=""
                         description="Tag id to be checked" displayName="Check
                         Tag Id"
                         encrypted="false">
        <valueType type="string"/>
      </Parameter>
      <Parameter id="71" name="TagCheckInterval" defaultValue="60000"
                         description="Time interval in milliseconds between two
                         tag-checking window"
                         displayName="Tag Check Interval" encrypted="false">
        <valueType type="int"/>
      </Parameter>
      <Parameter id="72" name="TagCheckTimeWindow" defaultValue="60000"
                         description="Time window in milliseconds for each
                         tag-checking"
                         displayName="Tag Check Time Window"
                         encrypted="false">
        <valueType type="int"/>
      </Parameter>
    </Parameters>
    <Version>1.0</Version>
  </Filter>
  <Filter id="73">
    <ClassName>oracle.edge.impl.filter.DebugFilter</ClassName>
    <Description>Debug Filter</Description>
    <Name>Debug Filter</Name>
    <Parameters id="74">
      <Parameter id="75" name="EventOutputFile" defaultValue=""
                         description="Output file for dumping events"
                         displayName="Debug Output File"
                         encrypted="false">
        <valueType type="string"/>
      </Parameter>
    </Parameters>
    <Version>1.0</Version>
  </Filter>
</FilterList>
```

Because a filter object is not executed and therefore cannot be used by a device, you must a create an instance of the filter (a filter instance) to enable a device to use the filter.

3.1.1 Device- and Device Group-Level Filtering

Filters can be attached to either a specific device or to a device group. Some filters are written as group-level filters and can only be attached to a device group. Likewise, some filters are written only for device-level filtering and only function if they are attached to a specific device. The filter object implements three levels of filtering:

- Pre-Device Filtering
- Post-Device Filtering
- Device Group Filtering

Pre-Device Filtering

Pre-device filtering provides filtering against a batch of pass-in events before they are routed to the Oracle Sensor Edge Server device.

Post-Device Filtering

Post-device filtering provides any filtering against the events before they are merged up to a device group.

Device Group Filtering

Device group filtering provides filtering against events before they are delivered to an edge client.

Note: Pre- and post-device filters apply only to devices; device group filtering applies only to device groups.

The Oracle Sensor Edge Server enables you to add filters that provide pre-device and post-device filtering. For more information on device group- and device-level filtering, refer to Section 3.3.1.

Note: Only the filters that you create enable pre- and post-device filtering. For more information on developing filters, refer to the *Oracle Application Server Wireless Developer's Guide.*

3.2 Defining the Parameters of the Pre-Seeded Filters

The Oracle Sensor Edge Server provides a set of pre-seeded filters (described in Table 3–1). If needed, you can develop your own filter extensions and then upload them. For more information on uploading extensions, refer to Chapter 4.

Table 3–1 The Pre-Seeded Filters of the Oracle Sensor Edge Server

Filter Name	Function	Applied to Device Group? (Supports Group-Level Filtering)	Applied to Devices? (Supports Device-Level Filtering)
Check Tag ID Filter	A diagnostic tool that checks if a device is reading tags during a specified interval. See Section 3.2.1	No	Yes
Cross-Reader Redundant Filter	Blocks redundant events that are sent from the devices of a device group. See Section 3.2.2	Yes	No
Debug Filter	Tracks events passing through the system and then writes these events to a log file. See Section 3.2.3	No	Yes

Table 3–1 (Cont.) The Pre-Seeded Filters of the Oracle Sensor Edge Server

Filter Name	Function	Applied to Device Group? (Supports Group-Level Filtering)	Applied to Devices? (Supports Device-Level Filtering)
Pass Filter	Notifies applications that a tag has passed through a device's field. This filter also blocks events, so that only one event, rather than duplicate events, is generated when a tag is detected by a device. See Section 3.2.4	No	Yes
Shelf Filter	Signals that an item has entered, or exited the field or gateway of a device reader. See Section 3.2.5	No	Yes
Pallet Pass-Thru Filter	Enables you to see all of the tag IDs for items held in a container or on a pallet. See Section 3.2.6	No	Yes
Pallet Shelf Filter	Sends events that signal new containers or pallets entering or exiting the field or gateway of a device reader See Section 3.2.7	No	Yes

The following sections describe how the pre-seeded filters generate events and their configuration parameters:

- Section 3.2.1, "Configuring the Check Tag ID Filter"
- Section 3.2.2, "Using the Cross-Reader Redundant Filter"
- Section 3.2.3, "Using the Debug Filter"
- Section 3.2.4, "Configuring the Pass Filter"
- Section 3.2.5, "Configuring the Shelf Filter"
- Section 3.2.6, "Configuring the Pallet Pass Thru Filter"
- Section 3.2.7, "Configuring the Pallet Shelf Filter"

3.2.1 Configuring the Check Tag ID Filter

A Check Tag is any normal tag used to test if the device (in this case, a reader) is reading tags. Because the Check Tag itself should be physically located within the field of the reader, it should always be read; when other tags move through the field of the reader, the device also reads the Check Tag in conjunction with them.

The Check Tag ID Filter ensures that the device reads a Check Tag periodically. Using this filter enables you to check the status of a driver, its corresponding reader, and attached antennae. Because the Check Tag ID Filter is used solely for diagnostic purposes, it does not provide any events for dispatching to client devices. Instead, this filter generates an event if it does not detect that the device has read a Check Tag for a specified period of time.

Note: You can apply the Check Tag ID Filter only to devices.

Table 3–2 describes the parameters (and associated values) of the Check Tag ID filter.

Table 3–2 Parameters of the Check Tag ID Filter

Name	Value Type	Description
CheckTagId	A `String` value.	The tag ID of the Check Tag, which is the ID that the filter searches for to see if the tag is being read.
TagCheckTimeWindow	An `int` value.	The period of time, in milliseconds, after which an event is generated if the filter has not seen the specified Check Tag.

To define the parameters for the Check Tag ID Filter, you must note the ID of the Check Tag itself (which must be placed within the field of the reading device). Enter this ID as the `String` value of `<CheckTagId>`. If the filter does not detect that a device has read a Check Tag bearing the specified ID for the period defined in the `<TagCheckTimeWindow>` parameter, it generates an event. Table 3–3 describes the signature of the generated event. Refer to Section 1.1.2 for more information on the Event type.

3.2.2 Using the Cross-Reader Redundant Filter

The Cross-Reader Redundant Filter blocks redundant events that are sent from the devices of a device group and does not generate any events. This filter considers events redundant if it finds they have the same tag ID.

The Cross-Reader Redundant Filter is for group-level filtering only; it performs no functions if applied to a device. This filter has no parameters to configure.

3.2.3 Using the Debug Filter

The Debug Filter traces events passing through the system. Upon receiving events from its associated device, this filter writes events to a log file. This filter has a single parameter called `<EventOutputFile>`. To define this parameter, enter the full path of the log file to which the Debug Filter writes events. (The server must make this file writable.) The format of the Debug Filter's output is:

```
"Devicename: <devicename> Type: <type> Subtype: <subtype>
EventTime: <time>TagIds:<tagid(,tagid)*>Data:<dat(,data)*>\n"
```

Each event is on a separate line; each line is separated by a newline character (LF or CRLF, depending on the operating system). The `<time>` value is a `long` as returned by the `time(2)` call.

This filter can only be attached to a device, not to a device group.

3.2.4 Configuring the Pass Filter

When a tag passes through the range of transmission, or through the gateway of a device reader, it generates a series of "tag is seen" events. The device reports these events periodically, starting when the tag enters the transmission field. The reporting stops when the tag moves out of the reader field.

Applications often do not require the series of events that a device reader generates; instead, these applications only need to know that a tag has passed through a device's gateway or range of transmission. The Pass Filter applies to such situations, as it reduces all of the "tag is seen" events into single events for each unique tag that passes through the field of a reader device.

The Pass Filter has one parameter, `<ExitEventThresholdTime>`. To define this parameter, enter the time (an `int` value), in milliseconds, since the device last read the tag before it is considered to have moved out of the device's transmission field. The parameter settings, which range from 50 milliseconds to under two seconds, dictate the frequency (that is, the reader cycle) in which the device reports these "tag is seen" events. If you set this frequency too high, such as to two seconds, then the device may miss the tag altogether.

When the device first reads a tag, the Pass Filter caches the tag's ID (tagid), notes the time that the tagid was read into the cache, and then immediately sends the pass-through event. The filter blocks subsequent reads for this cached tagid. Each time the filter receives a new read from the device, it updates the time that it read the tagid into the cache. If the sum of the caching time and the value set for `<ExitEventThresholdTime>` is less than the current time, then the Pass Filter clears the tagid from the cache. The next time the device reads this tag, the filter considers it a new event, caches its tagid and sends out a new pass-through event. Refer to Section 1.1.2 for more information on the Event type.

Table 3–3 describes the signature of the pass-through event.

Table 3–3 Signature of the Pass-Through Event

Event Field	Value
sourceName	This field identifies the originator of the event. This is an optional field.
correlationId	The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message sent back by the client has the same ID. This is an optional field.
siteName	The name of the site that generated this event.
deviceName	The name of the device that generated this event.
time	The time that the event was generated.
type	Event.OBSERVATION
subtype	Event.PASS
id	The ID of the tag.
data	The data payload of the tag.

3.2.5 Configuring the Shelf Filter

The Shelf Filter is a device-level filter that generates events when a tag is detected within the field of a reader or when the tag has left the field. Like the Pass Filter, the Shelf Filter has a single parameter, `<ExitEventThresholdTime>`. To define this parameter, enter the time (an `int` value), in milliseconds, since the device last read the tag before it is considered to have moved out of the device's transmission field. Unlike the Pass Filter, however, the Shelf Filter silently clears its cache once the interval defined for the `<ExitEventThresholdTime>` parameter elapses and does not generate an event.

3.2.5.1 Events Generated by the Shelf Filter

The Shelf Filter generates two events:

- IN FIELD Event
- OUT FIELD Event

IN FIELD Event

The Shelf Filter generates this event when the device first detects the tag. Table 3–4 describes the signature for this event. Refer to Section 1.1.2 for more information on the Event type.

Table 3–4 Signature of the IN FIELD Event

Event Field	Value
sourceName	This field identifies the originator of the event. This is an optional field.
correlationId	The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message sent back by the client has the same ID. This is an optional field.
siteName	The name of the site that generated this event.
deviceName	The name of the device that generated this event.
time	The time that the Shelf Filter generated this event.
type	Event.OBSERVATION
subtype	Event.INFIELD
id	The ID of the tag.
data	The data payload of the tag.

OUT FIELD Event

The Shelf Filter generates this event when the interval defined for the *Exit Event Threshold Time* parameter elapses. Table 3–5 describes the signature for this event. Refer to Section 1.1.2 for more information on the Event type.

Table 3–5 Signature of the OUT FIELD Event

Event Field	Value
sourceName	This field identifies the originator of the event. This is an optional field.
correlationId	The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message sent back by the client has the same ID. This is an optional field.
siteName	The name of the site that generated this event.

Table 3–5 (Cont.) Signature of the OUT FIELD Event

Event Field	Value
deviceName	The name of the device that generated this event.
time	The time that the Shelf Filter generated this event.
type	Event.OBSERVATION
subtype	Event.OUTFIELD
id	The ID of the tag.
data	The data payload of the tag.

When a device first detects the tag, the Shelf Filter caches the ID of the tag and then generates an IN FIELD event. At this point, the tag is read during every reader cycle. While the tag may not be read during some of these cycles, it is read during others. When the device does not read the tag consistently for a period longer than that designated for the `<EventExitThresholdTime>` parameter, then the filter removes the tag's ID from the cache and generates an OUT FIELD event. The devices stops reading the tag once it passes from the field of the device.

3.2.6 Configuring the Pallet Pass Thru Filter

The Pallet Pass Thru Filter collects all of the events received during a specified period and sends them out as a single event. When a pallet or container passes through a gateway or through the field of transmission of a reader device, this filter generates a single event for all of these tags. This filter enables you to see what items a container or pallet may hold.

The Pallet Pass Thru Filter includes the following parameters:

- <ExitEventThresholdTime>
- <EventCollectControlTime>

<ExitEventThresholdTime>

To define this parameter, enter the time (an `int` value), in milliseconds, since the device last read a tag before it is considered to have moved out of the device's transmission field. The parameter settings, which range from 50 milliseconds to under two seconds, dictate the frequency (that is, the reader cycle) in which the device reports these "tag is seen" events. If you set this frequency too high, such as to two seconds, then the device may miss the tag altogether.

<EventCollectControlTime>

To define this parameter, enter the time (an `int` value), in milliseconds, for a device to complete a reading cycle of the tags included in a pallet or container before starting a new reading cycle. When this time elapses, the reading cycle concludes (that is, the device has read all of the new tags) and the Pallet Pass Thru Filter then generates an event with the following signature (described in Table 3–6). Refer to Section 1.1.2 for more information on the Event type.

Table 3–6 Signature of the Pallet Pass Thru Event

Event Field	Value
sourceName	This field identifies the originator of the event. This is an optional field.
correlationId	The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message sent back by the client has the same ID. This is an optional field.
siteName	The name of the site that generated this event.
deviceName	The name of the device that generated this event.
time	The time that the event was generated.
type	Event.OBSERVATION
subtype	Event.MULTIPLE_PASS
id	A comma-separated list of tag IDs.
data	A comma-separated list of datum.

3.2.7 Configuring the Pallet Shelf Filter

The Pallet Shelf Filter collects all of the events received during a set interval and then sends them as a single event. This filter enables you to identify when new containers or pallets holding many items enters an area, or exits the field or gateway of a device reader.

The Pallet Shelf Filter has the following parameters:

- <ExitEventThresholdTime>
- <EventCollectControlTime>

<ExitEventThresholdTime>

To define `<ExitEventThresholdTime>`, enter the time (an `int` value), in milliseconds, from the last time that the device read the tag before it is considered to have moved out of the device's transmission field. The Pallet Shelf Filter silently clears its cache once the interval defined for the `<ExitEventThresholdTime>` parameter elapses and does not generate an event.

<EventCollectControlTime>

To define `<EventCollectControlTime>`, enter the time (an `int` value), in milliseconds, for a device to complete a reading cycle for the tags of a pallet or container before starting a new reading cycle. When this time elapses, the reading cycle concludes (that is, the device has read all of the new tags) and the Pallet Shelf Filter then generates an event.

3.2.7.1 Events Generated by the Pallet Shelf Filter

The Pallet Shelf Filter generates two events:

- MULTIPLE IN FIELD Event
- MULTIPLE OUT FIELD Event

MULTIPLE IN FIELD Event

The Pallet Shelf Filter generates the MULTIPLE IN FIELD event when the device first detects the tags. This event has the following signature (described in Table 3–7).

Table 3–7 Signature of the MULTIPLE IN FIELD Event

Event Field	Value
sourceName	This field identifies the originator of the event. This is an optional field.
correlationId	The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message sent back by the client has the same ID. This is an optional field.
siteName	The name of the site that generated this event.
deviceName	The name of the device reading the pallet or container that generated this event.
time	The time that the Pallet Shelf Filter generated this event.
type	Event.OBSERVATION
subtype	Event.MULTIPLE_INFIELD
id	A comma-separated list of tag IDs.
data	A comma-separated list of datum.

MULTIPLE OUT FIELD Event

The Pallet Shelf Filter generates the MULTIPLE OUT FIELD event when the interval defined for the `<ExitEventThresholdTime>` parameter elapses. This event has the following signature (described in Table 3–8):

Table 3–8 Signature of the MULTIPLE OUT FIELD Event

Event Field	Value
sourceName	This field identifies the originator of the event. This is an optional field.
correlationId	The client sets the value for this field, which is used for message responses to a particular client (such as checking if a device functions). Any message sent back by the client has the same ID. This is an optional field.
siteName	The name of the site generating this event.
deviceName	The name of the device reading the pallet or container that generated this event.
time	The time that the Pallet Shelf Filter generated this event.
type	Event.OBSERVATION

Table 3–8 (Cont.) Signature of the MULTIPLE OUT FIELD Event

Event Field	Value
subtype	Event.MULTIPLE_OUTFIELD
id	A comma-separated list of Tag IDs.
data	A comma-separated list of datum.

3.3 Enabling Event Filtering for Devices or Device Groups

Filter instances enable events to be filtered for both devices and device groups. To enable filtering, you create a filter instance from a filter object defined in the edgeserver.xml file's `<FilterList>` element (Example 3–1). You define these filter instances (that is, the `<FilterInsts>` tags) within the `<DeviceGroups>` elements of `edgeserver.xml` (illustrated in Example 3–2).

3.3.1 Creating a Filter Instance

To create a filter instance:

1. Within the `<DeviceGroups>`, locate the `<Device>` to which to add the filter instance.
2. Add a `<FilterInst>` element to the device's `<FilterInsts>` section.
3. For the `<Extension>` element, specify the name of the filter for the `class` attribute, and the `id` of the filter for the `reference` attribute.
4. Set the `<ParameterInsts>` to match the attribute values set for the filter's `<Parameters>` element.
5. Save `edgeserver.xml` and then restart the Oracle Sensor Edge Server. For more information on starting and stopping the Oracle Sensor Edge Server, refer to Section 1.3.

Note: You define device-level filters using the `<FilterInsts>` tag of the `<Device>` element. You define device group-level filters using the `<Filterinsts>` tag of the `<DeviceGroup>` element.

3.3.1.1 Prioritizing Filter Instances for Devices and Device Groups

You can stack filters together. For example, you can configure a device to first filter out redundant events and then aggregate them with another custom filter. While you can stack both of these filters on top of the same device, you must prioritize these filters, as the order in which the device's filters process events is crucial; the proper filter sequence for the aforementioned device would not first aggregate events and then filter for the redundant events.

The `<Sequence>` tag within the `<FilterInsts>` element for a device sets the order in which filters are invoked when a device reads an event. This sequence ranges from 0 to 65535. The filter assigned the lowest value in the sequence is placed closest to the data source (a device or a device group). The filters process the events according to the numbers that you assign to them. After a device reads an event, the filter with the lowest number first processes the event and then passes the output onto the filter assigned to the next number in the sequence.

If two or more filters have the same sequence number, then the order in which the filters process events is random. For example, there are five filters with sequence

numbers of 0, 1, 2, 3, 3, 4 set for a device. When the device receives an event, it first passes it to the filter assigned with 0 (Filter 0). Filter 0 processes the event and then passes the output to the next filter, Filter 1, which in turn processes the output from Filter 0 and passes its own output on to Filter 2, which continues the filtering cycle and passes its output to Filter 3. Because the device has two filters that are assigned the same value in the sequence, The Oracle Edge Sensor Server selects one of these filters at random to first process the output generated by Filter 2. When the next Filter 3 completes its task, it passes its output to Filter 4.

Example 3–2 illustrates configuring a device to use a filter instance by defining the <FilterInsts> element. In this example, the PalletPassFilter has a <Sequence> value set at 1.

Example 3–2 Configuring a Filter Instance for a Device

```
<FilterInsts>
      <FilterInst>
          <Sequence>1</Sequence>
          <FilterName>PalletPassFilter</FilterName>
          <Extension class="Filter" reference="50"/>
          <ParameterInsts>
            <ParameterInst>
              <ParameterMetaData reference="52"/>
              <Value>11</Value>
              <Name>ExitEventThresholdTime</Name>
            </ParameterInst>
            <ParameterInst>
              <ParameterMetaData reference="53"/>
              <Value>12</Value>
              <Name>EventCollectControlTime</Name>
            </ParameterInst>
          </ParameterInsts>
          <NeedReload>false</NeedReload>
          <Name>PalletPassFilter</Name>
        </FilterInst>
 </FilterInsts>
```

4

Managing Extensions

This chapter, through the following sections, describes the extensions for the Oracle Sensor Edge Server's driver, filters, and dispatchers.

4.1 Overview of Extensions

An extension is a custom-built driver, dispatcher or filter which you upload to the Oracle Sensor Edge Server by packaging the component in an Extension Archive file. The Extension Archive file is a JAR file containing all of the class files and native binaries for the driver, filter, or dispatcher, as well as properties files or static data files. In addition, the Extension Archive includes the Extension Archive Descriptor file, an XML file containing instructions for the Oracle Sensor Edge Server on loading and managing the extension.

Note: Setting the element content of `<IsExtensionMonitorEnabled>` to `true` enables an extension to be dynamically uploaded to the Oracle Sensor Edge Server. You do not have have to restart the Oracle Sensor Edge Server. However, for the Oracle Edge Sensor Server to use the instances created from an extension, you must restart the Oracle Edge Sensor Server as described in Section 1.3.1..

For information on the packaging an Extension Archive and the Device Management API, refer to the ***Oracle Application Server Wireless Developer's Guide***.

4.2 Extension Archive Files

Before you can upload a custom extension, such as a driver, dispatcher, or filter, you must package the extension files into an Extension Archive. An Extension Archive contains all of the extension's binaries, startup data, and configuration information. Each Extension Archive contains only one extension implementation, which is loaded at runtime. The Extension Archive contains the following directories:

- Meta-INF
- classes
- lib

Meta-INF

This directory contains any meta information about the archive. This directory must include the Extension Archive Descriptor file. The Extension Archive Descriptor file is an XML file located in the `META-INF` directory that contains the information needed by the Oracle Sensor Edge Server to load and manage the extension.

The Extension Archive Descriptor file is called ext.xml. Example 4–1 illustrates an Extension Archive Descriptor file (`ext.xml`) for a filter extension called Loop Back Filter.

Example 4–1 The Extension Archive Descriptor File for a Filter Extension

```
<?xml version="1.0"?>
<Extension>
<name>Loop Back Filter</name>
<version>1.0</version>
<className>oracle.edge.impl.filter.LoopBackFilter</className>
<type>Filter</type>
<Parameters>
        <Parameter name="TagID" defaultValue="" description="The Invalid Tag ID">
            <valueType type="string"/>
        </Parameter>
        <Parameter name="LightStackName" defaultValue="stack1" description="The
Light Stack Instance Name">
            <valueType type="string"/>
        </Parameter>
    </Parameters>
```

Example 4–2 describes a simplified version of the DTD for ext.xml; Table 4–1 describes this DTD's elements.

Example 4–2 The DTD for the Extension Archive Descriptor File

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Extension (name, version, className, type, Parameters)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT className (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT Parameters (Parameter+)>
<!ELEMENT Parameter (valueType)>
<!ATTLIST Parameter
  name CDATA #REQUIRED
  displayName CDATA #IMPLIED
  defaultValue CDATA #IMPLIED
  description CDATA #IMPLIED
  encrypted (true|false)  #IMPLIED
  isClearText (true|false) #IMPLIED
  required  (true|false)  #IMPLIED>
<!ELEMENT valueType EMPTY>
<!ATTLIST valueType
 type (int | string | double | boolean) #REQUIRED>
```

Table 4–1 Elements and Attributes of the DTD for the Extension Archive Descriptor File

Element	Attribute or Text	Description
`Extension`		Defines the properties of an extension.
`name`	`#text`	The name of the extension.
`type`	`#text`	The type of the extension, such as a driver, filter, or dispatcher. Although the match is not case-sensitive, there must be no extra spaces or special characters in the text. The reserved values are: Device, Filter, Dispatcher.
`version`	`#text`	A text representation of the version number of the extension.
`className`	`#text`	The name of the class to load and instantiate the driver. This is the entry class that implements one of the standard extension interfaces. You must include a package name to form a fully qualified class name.
`Parameters`	(Parent of the `<Parameter>` element.)	The parameters that users can edit after an extension has been uploaded.
`Parameter`	Attributes include: ■ `name` ■ `displayName` ■ `defaultValue` ■ `encrypted` ■ `isClearText` ■ `required`	■ `name` -- The name of the parameter. ■ `displayName` -- The display name of the parameter. ■ `defaultValue` -- The default value for the parameter. ■ `encrypted` -- Indicates whether the value for the parameter should be encrypted so that the value does not have to be stored in clear-text format. ■ `isClearText` -- Enables the default value (and the value for the parameter instance) to be reset to clear-text format. If the encrypted parameter is set to `true`, then the clear-text format is read and then set to encrypted format the next time the Oracle Sensor Edge Server starts. ■ `required` -- Indicates whether the parameter value is required.
`valueType`	`type`	The type of the parameter (which can be one of the following): ■ `int` -- if the parameter is a 32-bit signed integer. ■ `string` -- for a string of variable length. ■ `double` -- for a double precision number. ■ `boolean` -- for a boolean value (true, false).

classes

This directory includes all of the classes files, native binaries, files, or static data. The classes files packaged into JAR files must be expanded on top of this directory. This release does not support loading JAR libraries.

lib

The Extension Archive file also includes the `lib` directory, where you specify third-party libraries. Example 4–3 illustrates an Extension Archive file for an Alien device driver, where the `lib` directory includes the library specific to the Alien device driver, `Gateway.jar`.

Example 4–3 Extension Archive File for an Alien Device Driver

```
meta-inf/ext.xml
meta-inf/Manifest.mf
classes/oracle/edge/impl/driver/AlienReader.class
lib/Gateway.jar
```

4.2.1 Packaging an Extension Archive File

To package an Extension Archive file:

1. Build a sandbox directory. Use this directory as the JAR source directory.
2. At the top of this directory, create the `META-INF` and `classes` directories.
3. Copy all class files and properties files (if any) to the classes directory. In the META-INF directory, create `ext.xml`, the Extension Archive Descriptor file.
4. Archive the files. You can use the JAR tool included in the JDK, or any other standard compression utility. Run the JAR tool from top-level directory of the sandbox. For example, executing `jar cvMf test.jar` archives the files in the sandbox directory into `test.jar`. You can then upload `test.jar` to the Oracle Sensor Edge Server. Do not archive the `META-INF` and `classes` directories as part of the sandbox directory. For example, the command `c:/work> jar tvf test.jar` displays the files in `test.jar` have been properly archived as follows:

   ```
   0 Thu Apr 08 14:36:56 PDT 2004 META-INF/
   71 Thu Apr 08 14:36:56 PDT 2004 META-INF/ext.xml
   0 Thu Apr 08 13:42:52 PDT 2004 classes/
   0 Thu Apr 08 13:42:52 PDT 2004 classes/my/
   0 Thu Apr 08 13:42:58 PDT 2004 classes/my/test.class
   ```

Note: No slashes or other directory indicators appear before the `META-INF` and `classes` directories. Including the entire path in the JAR prevents the Oracle Sensor Edge Server from locating the Extension Archive Descriptor file or the classes. As a result, the extension cannot be deployed.

4.3 Uploading Extensions

To upload an extension:

1. Package the driver, filter, or dispatcher in an Extension Archive File as described in Section 4.2.1.
2. Copy this JAR file to:

   `ORACLE_HOME/edge/extensions`
3. Restart the Oracle Sensor Edge Server by restarting the OC4J Instance.

Tip: If the <IsExtensionMonitorEnabled> element has been set to true in edgeserver.xml, then you only need to copy the JAR file to ORACLE_HOME/edge/extensions. The running Oracle Sensor Edge Server then picks up the extension automatically and does not need to be stopped and then restarted. Because this method of adding an extension slows performance, it is recommended only for development instances.

4.4 Extension Class Hierarchy

All of the extensions of the Oracle Sensor Edge Server are arranged as:

EdgeObject—The basic root class, which contains a unique identifier

- AbstractEdgeExtensionImpl—Implements the EdgeExtension interface.
  - AbstractDispatcher—An abstract class that defines a base for a filter extension.
  - AbstractFilter—A basic filter.
  - AbstractDevice—A basic device.
    - AbstractEventDevice—A common sensor or indicator device.
    - AbstractSocketDevice—A network-based device driver.

Note: Extend from AbstractDispatcher, AbstractFilter, AbstractDevice (or from a descendant class) when you create an extension.

4.5 Implementing Extensions

The doInit() method implements extensions, as this call initializes the instance of an extension at runtime.

4.5.1 Extension Context

When the instance of an extension is created at runtime, the corresponding Context is created that enables the extension to:

- Set (or retrieve) the runtime Context data.
- Locate and communicate with other extensions of the Oracle Edge Sensor Server.
- Access the system facilities of the Oracle Edge Sensor Server.
- Retrieve information about the instance itself (described in Section 4.5.1.1).

4.5.1.1 Retrieving Information About the Instance

The base class, EdgeExtension, provides utility functions for an instance to retrieve information about itself. These methods include:

- getContext()

  Returns the runtime context.

- getName()

  Returns the name of the extension.

- `getDescription()`

  Returns the description of the extension.

- `getVersion()`

  Returns the version string of the extension.

4.5.1.2 Accessing the Runtime Context of an Instance

To retrieve the instance's `Context` object, use

```
EdgeExtensionContext context = super.getContext();
```

The method call, `getContext()`, returns the `Context` object of the current instance.

4.6 Managing the Parameters of an Instance

An instance of an extension does not hold its own persistent data or configuration; configuration data is passed in at runtime when the instance is initialized. The configuration data is defined as parameters, which are composed of name/value pairs. Each parameter has a unique name and an optional value.

Note: This release of the Oracle Sensor Edge Server does not directly support trees or arrays of values. You are responsible for un-marshalling the data when forming non-scalar type data.

4.6.1 Exposing Custom Parameters

Extensions often have specific configurations. For example, a driver might include such configuration parameters as serial port name, baud rate, IP address, port number, login and password. These parameters must be defined to enable the driver to communicate with the device.

To expose parameters for a driver implementation, you must modify the Extension Archive Descriptor file. Example 4–4 illustrates a device that has two parameters that can be configured: serial port name and baud rate, defined within the `<Parameter>` extract tags.

Example 4–4 An Extension Archive Descriptor File with Exposed Parameters

```
<Extension>
         <name>My Driver</name>
         <type>Device</type>
         <className>my.testdriver</className>
         <Parameters>
                   <Parameter name="port" displayName="Serial Port">
                          <valueType type="string"/>
                   </Parameter>
                   <Parameter name="baud" displayName="Baud Rate">
                         <valueType type="int"/>
                   </Parameter>
         </Parameters>
</Extension>
```

4.6.2 Retrieving Parameter Values

Once you have defined the Extension Archive Descriptor file's `<Parameter>` tags, you can fetch the values for the parameters using the `EdgeExtensionConfigInfo`

object. The values defined within the <Parameter> tags are retrieved using the Context object (illustrated in Example 4–5).

Example 4–5 Retrieving Parameter Values Using the Context Object

```
EdgeExtensionContext context = super.getContext();
ConfigParameter filenameParam = ct.getParameter( fileName );
```

The getParameter() method returns a ConfigParameter object. The getParameter() method returns the value for a parameter. (In Example 4–5, the ConfigParamter object is called filenameParam and the getParameter() method returns the value for a parameter called fileName.)The name of the target parameter must be passed to the ConfigParameter object. Further, the name of this parameter must match the name given to the name attribute of the <Parameter> element of the Extension Archive Descriptor file. Once you obtain the ConfigParameter object, you can get the value of the parameter (illustrated in Example 4–6).

Example 4–6 Retrieving the Value of a Parameter

```
m_fileName = filenameParam.getStringValue();
```

Note: The getStringValue()method returns the string value of the parameter. If the value for the parameter is an int, call the getIntegerValue() method, which returns an Integer object.

A

Sample edgeserver.xml File

This appendix describes the configuration file for the Oracle Sensor Edge Sever, `edgeserver.xml`.

A.1 edgeserver.xml

Example A–1 illustrates `edgeserver.xml` with comments. In this example, the configuration connects an Intermec device to the Oracle Sensor Edge Server, which disaptches events using the Oracle Streams dispatcher.

Example A–1 edgeserver.xml

```
<EdgeServer id="1">
  <Name>MyEdgeServer</Name>
  <SiteName>MySite</SiteName>
  <DispatcherMode>persist</DispatcherMode>
  <IsRunJmx>false</IsRunJmx>
  <JmxConsolePort>8989</JmxConsolePort>
  <JmxConsolePassword>welcome</JmxConsolePassword>
  <IsExtensionMonitorEnabled>false</IsExtensionMonitorEnabled>
  <LogLevel>notify</LogLevel>
  <ShutDownTimeOut>10000</ShutDownTimeOut>
  <DispatcherList id="2">
    <Dispatcher id="3">
      <ClassName>oracle.edge.rt.NullDispatcher</ClassName>
      <Description>Dispatcher that does nothing</Description>
      <Name>NullDispatcher</Name>
      <Parameters id="4"/>
      <Version>1.0</Version>
    </Dispatcher>
    <Dispatcher id="5">
      <ClassName>oracle.edge.impl.dispatcher.StreamsConnector</ClassName>
      <Description>StreamsDispatcher</Description>
      <Name>StreamsDispatcher</Name>
      <Parameters id="6">
        <Parameter id="7" name="username" defaultValue="edge"
                          description="stream access user name"
                          displayName="DBUsername"
                          encrypted="false">
          <valueType type="string"/>
        </Parameter>
                  --**CHANGED default value for password
        <Parameter id="8" name="password" defaultValue="oracle"
                          description="stream access password"
                          displayName="DB password" encrypted="false">
          <valueType type="string"/>
```

```
        </Parameter>
                  --**CHANGED value for URL to machine name PORT to 9105 and SID
                  to PRJ1
        <Parameter id="9"
name="url"defaultValue="jdbc:oracle:thin:@(description=(address=(host=soclxs3db02)

defaultValue="jdbc:oracle:thin:@(description=(address=(host=soclxs3db02)

(protocol=tcp)(port=9105))(connect_data=(sid=PRJ1)))"
                                              description="stream access url"
displayName="DB Connect String"
                                             encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Dispatcher>
    <Dispatcher id="10">
      <ClassName>oracle.edge.impl.dispatcher.HTTPEventDispatcher</ClassName>
      <Description>Http Dispatcher</Description>
      <Name>HTTP Dispatcher</Name>
      <Parameters id="11">
        <Parameter id="12" name="url"
defaultValue="http://localhost:8888/rfid/test"
                                               description="HTTP url"
encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="13" name="disableAcceptCookieDialog" defaultValue="true"
                           description="Whether to disable the accept/reject
                           cookie dialog from HttpClient"
                           encrypted="false">
          <valueType type="boolean"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Dispatcher>
    <Dispatcher id="14">
      <ClassName>oracle.edge.impl.dispatcher.JMSEventDispatcher</ClassName>
      <Description>JMS Dispatcher</Description>
      <Name>JMS Dispatcher</Name>
      <Parameters id="15">
        <Parameter id="16" name="provider" defaultValue="ormi://localhost"
                           description="JMS provider url" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="17" name="username" defaultValue="admin"
                           description="JMS user name" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="18" name="password" defaultValue="mobile"
                           description="JMS user password" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="19" name="ack" defaultValue="CLIENT_ACKNOWLEDGE"
                           description="JMS acknowledge mode" encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
```

```
    </Dispatcher>
    <Dispatcher id="20">
      <ClassName>oracle.edge.impl.dispatcher.WSEventDispatcher</ClassName>
      <Description>WebService Dispatcher</Description>
      <Name>WebService Dispatcher</Name>
      <Parameters id="21">
        <Parameter id="22" name="url" defaultValue="http://localhost:8888/wsdl/"
                           description="url to locate wsdl" encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Dispatcher>
  </DispatcherList>
  <DriverList id="23">
    <Driver id="24">
      <ClassName>oracle.edge.impl.driver.EdgeSimulator</ClassName>
      <Description>This is internal simulator</Description>
      <Name>Edge Simulator Driver</Name>
      <Parameters id="25">
        <Parameter id="26" name="FileName"
defaultValue="..\..\edge\config\Simulation.xml"
                           description="Simulator's configuration file"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
    <Driver id="27">
      <ClassName>oracle.edge.impl.driver.AlienReader</ClassName>
      <Description>This is an alien device</Description>
      <Name>AlienDevice</Name>
      <Parameters id="28">
        <Parameter id="29" name="PortNo" defaultValue="23"
                           description="Alien reader's open port number that edge
                           device listens to"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="30" name="IPAddress" defaultValue="144.25.171.23"
                                            description="Alien reader's IP
address" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="31" name="UserName" defaultValue="alien" description="Alien
                           reader's access user"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="32" name="Password" defaultValue="password"
                           description="Alien reader's access password"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="33" name="AntennaSeqIdList" defaultValue=""
                           description="List of identifiers to identify each
                           antenna" encrypted="false">
          <valueType type="string"/>
        </Parameter>
```

```
        <Parameter id="34" name="AntennaMappedDeviceNameList" defaultValue=""
                           description="List of mapped device names associated
                           with each antenna"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
    <Driver id="35">
      <ClassName>oracle.edge.impl.driver.IntermecReader</ClassName>
      <Description>This is Intermec reader: IntelliTag 500</Description>
      <Name>IntermecDevice</Name>
      <Parameters id="36">
                                --**CHANGED default value for PORTNO to 6543 (where
DeviceManager is listening
        <Parameter id="37" name="PortNo" defaultValue="6543"
                           description="Reader's open port number that edge device
                           listens to"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
                    --**CHANGED value for IPADDRESS (IP of machine RFID hardware is
connected to)
        <Parameter id="38" name="IPAddress" defaultValue="192.168.0.52"
                           description="Reader's IP address" encrypted="false">
                          <valueType type="string"/>
        </Parameter>
        <Parameter id="39" name="AntennaSeqIdList" defaultValue=""
                           description="List of identifiers to identify each
                           antenna" encrypted="false">
          <valueType type="string"/>
        </Parameter>
        <Parameter id="40" name="AntennaMappedDeviceNameList" defaultValue=""
                           description="List of mapped device names associated
                           with each antenna"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
    <Driver id="41">
      <ClassName>oracle.edge.impl.driver.EdgeEventDevice</ClassName>
      <Description>This is EMS reader.</Description>
      <Name>EMSDevice</Name>
      <Parameters id="42">
        <Parameter id="43" name="PortNo" defaultValue="6666"
                           description="Reader's open port number that edge device
                           listens to"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="44" name="IPAddress" defaultValue="144.25.168.131"
                           description="Reader's IP address" encrypted="false"
<valueType type="string"/>
        </Parameter>
        <Parameter id="45" name="AntennaSeqIdList" defaultValue=""
                           escription="List of identifiers to identify each
antenna" encrypted="false">
```

```
          <valueType type="string"/>
        </Parameter>
        <Parameter id="46" name="AntennaMappedDeviceNameList" defaultValue=""
                           description="List of mapped device names associated
                           with each antenna"
                           encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
    <Driver id="47">
      <ClassName>oracle.edge.impl.driver.EdgeDevice</ClassName>
      <Description>Edge Device Driver</Description>
      <Name>Edge Device Driver</Name>
      <Parameters id="48">
        <Parameter id="49" name="PortNo" defaultValue="23"
                           description="Edge device's open port number that edge
                           device listens to"
                           displayName="Port Number" encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="50" name="IPAddress" defaultValue="" description="Edge
                                          device's IP address"
                           displayName="IP Address" encrypted="false">
          <valueType type="string"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Driver>
  </DriverList>
  <FilterList id="51">
    <Filter id="52">
      <ClassName>oracle.edge.impl.filter.PassFilter</ClassName>
      <Description>Filter redundant in range tag ids from a single
reader.</Description>
      <Name>PassRedundantFilter</Name>
      <Parameters id="53">
        <Parameter id="54" name="ExitEventThresholdTime" defaultValue="800"
                           description="Time elapsed in milliseconds since a tag
                                        has been seen last time"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Filter>
    <Filter id="55">
      <ClassName>oracle.edge.impl.filter.PalletPassFilter</ClassName>
      <Description>Filter redundant in range tag ids from a single
reader.</Description>
      <Name>PalletPassFilter</Name>
      <Parameters id="56">
        <Parameter id="57" name="ExitEventThresholdTime" defaultValue="800"
                           description="Time elapsed in milliseconds since a tag
                                        has been seen last time"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="58" name="EventCollectControlTime" defaultValue="1500"
```

```
                                                description="Time elapsed in
milliseconds since a new tag has been detected last
time"
 encrypted="false">
          <valueType type="int"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Filter>
    <Filter id="59">
      <ClassName>oracle.edge.impl.filter.PalletShelfFilter</ClassName>
      <Description>Filter redundant in range tag ids from a single
reader.</Description>
      <Name>PalletShelfFilter</Name>
      <Parameters id="60">
        <Parameter id="61" name="ExitEventThresholdTime" defaultValue="800"
                           description="Time elapsed in milliseconds since a tag
                                        has been seen last time"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
        <Parameter id="62" name="EventCollectControlTime" defaultValue="1500"
                           description="Time elapsed in milliseconds since a new
                                        tag has been detected last time"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Filter>
    <Filter id="63">
      <ClassName>oracle.edge.impl.filter.ShelfFilter</ClassName>
      <Description>Filter redundant in range tag ids from a single
reader.</Description>
      <Name>ShelfRedundantFilter</Name>
      <Parameters id="64">
        <Parameter id="65" name="ExitEventThresholdTime" defaultValue="800"
                           description="Time elapsed in milliseconds since a tag
                           has been seen last time"
                           encrypted="false">
          <valueType type="int"/>
        </Parameter>
      </Parameters>
      <Version>1.0</Version>
    </Filter>
    <Filter id="66">
      <ClassName>oracle.edge.impl.filter.CrossReaderRedundantFilter</ClassName>
      <Description>Filter redundant tag ids from multiple readers.</Description>
      <Name>CrossReaderRedundantFilter</Name>
      <Parameters id="67"/>
      <Version>1.0</Version>
    </Filter>
    <Filter id="68">
      <ClassName>oracle.edge.impl.filter.CheckTagFilter</ClassName>
      <Description>Check Tag Filter</Description>
      <Name>Check Tag Filter</Name>
      <Parameters id="69">
        <Parameter id="70" name="CheckTagId" defaultValue=""
                           description="Tag id to be checked" displayName="Check
                           Tag Id"
```

```
                          encrypted="false">
            <valueType type="string"/>
          </Parameter>
          <Parameter id="71" name="TagCheckInterval" defaultValue="60000"
                          description="Time interval in milliseconds between two
                                       tag-checking window"
                          displayName="Tag Check Interval" encrypted="false">
            <valueType type="int"/>
          </Parameter>
          <Parameter id="72" name="TagCheckTimeWindow" defaultValue="60000"
                          description="Time window in milliseconds for each
                                       tag-checking"
                          displayName="Tag Check Time Window" encrypted="false">
            <valueType type="int"/>
          </Parameter>
        </Parameters>
        <Version>1.0</Version>
      </Filter>
      <Filter id="73">
        <ClassName>oracle.edge.impl.filter.DebugFilter</ClassName>
        <Description>Debug Filter</Description>
        <Name>Debug Filter</Name>
        <Parameters id="74">
          <Parameter id="75" name="EventOutputFile" defaultValue=""
                          description="Output file for dumping events"
                          displayName="Debug Output File"
                          encrypted="false">
            <valueType type="string"/>
          </Parameter>
        </Parameters>
        <Version>1.0</Version>
      </Filter>
    </FilterList>
    <CurrentDispatcher id="76">
                                --**CHANGED DispatcherName to StreamsDispatcher
      <DispatcherName>StreamsDispatcher</DispatcherName>
                                --**CHANGED reference to point to StreamsDispatcher
"5"
      <Extension class="Dispatcher" reference="5"/>
                                --**CHANGED default identify name to
StreamsDispatcher
      <Name>StreamsDispatcher</Name>
      <NeedReload>false</NeedReload>
                                --**ADDED ParameterInsts tag
      <ParameterInsts id="77">
                                --**ADDED ParameterInst tag for username
        <ParameterInst id="78">
          <Name>username</Name>
          <ParameterMetaData reference="7"/>
          <Value>edge</Value>
        </ParameterInst>
                        -**ADDED ParameterInst tag for password
        <ParameterInst id="79">
          <Name>password</Name>
          <ParameterMetaData reference="8"/>
          <Value>oracle</Value>
        </ParameterInst>
                        --**ADDED ParameterInst tag for URL
        <ParameterInst id="80">
          <Name>url</Name>
```

```
        <ParameterMetaData reference="9"/>
        <Value>jdbc:oracle:thin:@(description=(address=(host=soclxs3db02)
                      (protocol=tcp)(port=9105))(connect_data=(sid=PRJ1)))</Value>
      </ParameterInst>
    </ParameterInsts>
  </CurrentDispatcher>
  <DeviceGroups id="81">
    <DeviceGroup id="82">
      <DeviceList id="83">
                        --**ADDED Device tag for Intermec reader
        <Device id="84">
          <Name>Intermec Device</Name>
          <DriverName>IntermecDevice</DriverName>
                              --**NOTE make sure that extension reference points
to IntermecDevice listed above
          <Extension reference="35"/>
          <ParameterInsts id="85">
            <ParameterInst id="86">
              <Name>PortNo</Name>
                              --**NOTE make sure that extension reference points
to IntermecDevice listed above
              <ParameterMetaData reference="37"/>
              <Value>6543</Value>
            </ParameterInst>
            <ParameterInst id="87">
              <Name>IPAddress</Name>
                              --**NOTE make sure that extension reference points
to IntermecDevice listed above
              <ParameterMetaData reference="38"/>
              <Value>192.168.0.52</Value>
            </ParameterInst>
            <ParameterInst id="88">
              <Name>AntennaSeqIdList</Name>
                              --**NOTE make sure that extension reference points
to IntermecDevice listed above
              <ParameterMetaData reference="39"/>
              <Value>12000000</Value>
            </ParameterInst>
            <ParameterInst id="89">
              <Name>AntennaMappedDeviceNameList</Name>
                              --**NOTE make sure that extension reference points
to IntermecDevice listed above
              <ParameterMetaData reference="40"/>
              <Value>IT500_READER</Value>
            </ParameterInst>
          </ParameterInsts>
          <FilterInsts id="90"/>
        </Device>
      </DeviceList>
      <EventCollectWaitTime>500</EventCollectWaitTime>
      <FilterInsts id="91"/>
      <IsDefault>false</IsDefault>
      <IsSystem>true</IsSystem>
      <Name>Unassigned</Name>
    </DeviceGroup>
  </DeviceGroups>
</EdgeServer>
```

B

Troubleshooting

This appendix describes the following:

B.1 Error Messages after Start-Up

Problem

After configuration, the Oracle Sensor Edge Server displays error messages while starting up, such as:

```
End tag does not match start tag 'ParameterInst'
```

or

```
No such field
oracle.edge.jmx.model.DispatcherInstanceImpl.ParameterInst
```

Solution

An invalid XML document might cause this error. Check if the `edgeserver.xml` file is well-formed by opening the `edgeserver.xml` file with an XML editor or with Internet Explorer. In addition, check whether the reference values point to the correct ids and verify that all id values in the `edgeserver.xml` are unique.

B.2 Tag Reads Are Not Dispatched to the Back-End

Problem

The Oracle Sensor Edge Server starts properly, but the tag reads are not dispatched to the back-end.

Solution

Check the log file for any problems in starting the device.

B.3 Exceptions Thrown When Stopping the Oracle Sensor Edge Server

Problem

Exceptions are thrown when I stop the Oracle Sensor Edge Server.

Solution

Improper shut-down of the device controller might cause exceptions to be thrown. Stop the Oracle Sensor Edge Server before the device controller.

B.4 Additional Information About the Oracle Sensor Edge Server

Problem

I need additional information about the Oracle Sensor Edge Server. Where can I find more information and access the downloads? Is there a Web site specific to the Oracle Sensor Edge Server?

Solution

Oracle Technology Network (`http://www.oracle.com/technology/`) provides the lastest filters, drivers, and the Edge Development Kit for developing extensions to the Oracle Sensor Edge Server, as well as information and tutorials.

B.5 Installation Requirements

The Oracle Edge Sensor Server requires JDK 1.4.x and free ports.

Glossary

antenna

Each tag has at least one antenna. On the other side of the communication link, the reader must also have an antenna. Some readers can drive multiple antennae at the same time. Depending on the protocol, frequency and application, these antennae vary from thin strips of metal laid across a surface, to a portal doorway antenna that is meters tall

chip

A silicon chip, with embedded memory, is used in a tag. The chip implements the wireless protocol and access functions to its embedded memory. Note that in *Active Tags*, this is not a single chip but an entire board. See **tag**.

device

An end point of a sensor-based architecture, such as an RFID reader, a dry contact, a laser diode, carousel, or a robotic picker.

Oracle Sensor Edge Server

The server that resides between all of the readers and the application middle tier. It is responsible for interfacing with all of the readers and sending normalized data back to the application server.

Radio Frequency Identification (RFID)

RFID is the use of small transponders with embedded Electronic Serial Numbers (ESNs) or memory, which transmit identifiers across one or more frequencies.

reader

A reader reads from, and writes to, the tags to which it is connected. Readers usually have serial interfaces used to communicate with a host computer. There is no widely-accepted standard for this protocol. The process of retrieving data stored on an RFID tag by sending radio waves to the tag and then converting the waves the tag sends back into data is known as a read.

reader field

The area of coverage for a reader. If tags are outside of a reader field, then they cananot receive radio waves and cannot be read.

Real Time Location System (RTLS)

A technology that uses radio-frequency to produce real-time location information for tagged items.

tag

(Also known as an RFID tag.) A single unit that contains a chip, one or more antennae, and a power source. If it is battery-driven or from a external source, the tag is an *Active Tag*. If the power source is inductive-based (which means that it relies on photoelectric effect to generate power from remotely generated radio waves), the tag is a *Passive Tag*. A tag containing data that cannot be changed is a read-only tag. See **chip**.

Index