February 2015
Contents

Preface ................................................................................................................................................................ vii

Audience ............................................................................................................................................................... vii
Documentation Accessibility ................................................................................................................................. vii
Related Documentation ........................................................................................................................................ vii
Conventions ........................................................................................................................................................ vii

1 Building a Ticketing Connector

1.1 Introduction ...................................................................................................................................................... 1-1
1.2 How a Ticketing Connector Functions ........................................................................................................... 1-2
1.3 Prerequisites ..................................................................................................................................................... 1-4
1.4 Extracting Schema Files ................................................................................................................................ 1-4
1.5 Building a Ticketing Connector ......................................................................................................................... 1-4
1.5.1 Determining Connector Functionality ......................................................................................................... 1-4
1.5.2 Developing Required Template Files ......................................................................................................... 1-6
1.5.2.1 getTicket Request Template .................................................................................................................. 1-6
1.5.2.2 getTicket Response Template .................................................................................................................. 1-7
1.5.2.3 Default Template(s).................................................................................................................................. 1-8
1.5.2.4 createTicket Response Template ........................................................................................................... 1-9
1.5.2.5 publishTicket Request Template ............................................................................................................ 1-11
1.5.3 Defining the Connector Descriptor File ....................................................................................................... 1-12
1.5.3.1 Connector Deploy Field Information ..................................................................................................... 1-13
1.5.3.2 Connector Information Section .............................................................................................................. 1-13
1.5.3.3 Sample Connector Information Section ................................................................................................ 1-14
1.5.3.4 Authentication Section .......................................................................................................................... 1-14
1.5.3.5 Sample Authentication Section ............................................................................................................ 1-15
1.5.3.6 Connectivity Test Variable Section ....................................................................................................... 1-15
1.5.3.7 Sample Connectivity Test Variable Section .......................................................................................... 1-16
1.5.3.8 External URL Section ............................................................................................................................ 1-16
1.5.3.9 Sample External URL Section ................................................................................................................ 1-17
1.5.3.10 Service Section ..................................................................................................................................... 1-17
1.5.3.11 Sample Service Section ....................................................................................................................... 1-18
1.5.3.12 Template Registration Section ............................................................................................................ 1-19
1.5.3.13 Sample Template Registration Section .............................................................................................. 1-20
1.5.3.14 Complete Connector Deployment File ................................................................................................ 1-21
1.6 Packaging and Deploying the Ticketing Connector ......................................................................................... 1-21
2 Building an Event Connector

2.1 Introduction ............................................................................................................. 2-1
2.2 How an Event Connector Functions ...................................................................... 2-2
2.3 Prerequisites .......................................................................................................... 2-3
2.4 Extracting Schema Files ...................................................................................... 2-3
2.5 Building an Event Connector ................................................................................ 2-4
2.5.1 Determining Connector Functionality ............................................................... 2-4
2.5.2 Developing Required Template Files ............................................................... 2-5
2.5.2.1 setup/initialize/uninitialize/cleanup Request Template ................................ 2-6
2.5.2.2 setup/initialize/uninitialize Response Template ........................................... 2-6
2.5.2.3 createEvent/updateEvent Request Template .............................................. 2-7
2.5.2.4 createEvent/updateEvent Response Template .............................................. 2-8
2.5.3 Defining the Connector Descriptor File ............................................................ 2-9
2.5.3.1 Connector Deploy Field Information ............................................................ 2-10
2.5.3.2 Connector Information Section ................................................................... 2-10
2.5.3.3 Sample Connector Information Section ....................................................... 2-11
2.5.3.4 Authentication Section ............................................................................... 2-11
2.5.3.5 Sample Authentication Section .................................................................. 2-12
2.5.3.6 Service Section ............................................................................................ 2-12
2.5.3.7 Sample Service Section .............................................................................. 2-13
2.5.3.8 Template Registration Section ................................................................... 2-14
2.5.3.9 Sample Template Registration Section ....................................................... 2-15
2.5.3.10 Complete Connector Deployment Field .................................................... 2-16
2.6 Packaging and Deploying the Event Connector .................................................. 2-17

3 Building a Data Exchange Connector

3.1 Introduction ............................................................................................................. 3-1
3.1.1 Enterprise Manager and External Management System .................................... 3-2
3.1.2 Data Forwarding Frequency Options and Modes ............................................. 3-2
3.2 Data Exchange Concepts ...................................................................................... 3-3
3.2.1 Data Exchange Hub ......................................................................................... 3-3
3.2.2 Inbound Data Exchange Session .................................................................... 3-3
3.2.3 Outbound Data Exchange Session ................................................................... 3-3
3.2.3.1 Normalized Message Format ..................................................................... 3-4
3.2.3.2 Denormalized Message Format ................................................................. 3-4
3.2.4 Message Schemas ............................................................................................ 3-4
3.2.5 Data Source ..................................................................................................... 3-4
3.2.6 Average Data ................................................................................................... 3-5
3.3 Setting up a Data Exchange Connector .................................................................. 3-5
3.3.1 Creating a Data Exchange Hub ....................................................................... 3-5
3.3.2 Using a Third-Party JMS Server as a Data Exchange Hub ............................... 3-6
3.3.3 Creating an Outbound Data Exchange Session ............................................. 3-7
3.3.4 Outbound JMS Destinations ......................................................................... 3-8
3.3.5 Outbound Message Schema .......................................................................... 3-13
3.3.5.1 Normalized Message Format ..................................................................... 3-13
3.3.5.2 Denormalized Message Format ................................................................. 3-22
3.3.6 Tuning Outbound Session Message Parameters ............................................ 3-27
E Create Event Template Example
E.1 Study Data in External Application ................................................................. E-1
E.2 Study Data in Enterprise Manager ................................................................. E-1
E.3 Determine the Mapping for the Template ......................................................... E-1
E.4 Create the Template ......................................................................................... E-2
E.5 Create Mappings ............................................................................................. E-2
E.5.1 Mapping the Summary Field ......................................................................... E-2
E.5.2 Mapping the Description Field ...................................................................... E-2
E.5.3 Mapping the Severity Field .......................................................................... E-3
E.5.4 Mapping the Priority Field ........................................................................... E-3
E.6 Completed Template Examples ........................................................................ E-3

F Sample Incident Data
F.1 Sample Create Transaction ............................................................................. F-1
F.2 Sample Update Transaction ............................................................................. F-5
F.3 Sample Close Transaction ............................................................................... F-9

G Sample Event Data
G.1 Sample Create Transaction ............................................................................. G-1
G.2 Sample Update Transaction ............................................................................. G-4
G.3 Sample Close Transaction ............................................................................... G-7

Glossary

Index
Preface

This Oracle Enterprise Manager Connectors Integration Guide provides the information that you will need to build ticketing and event connectors for integration with Oracle Enterprise Manager.

Audience

This document is intended for system integrators who want to integrate other management systems with Enterprise Manager.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documentation

The latest versions of this and other Oracle Enterprise Manager documentation can be found at:

http://docs.oracle.com/cd/E24628_01/index.htm

Oracle Enterprise Manager also provides extensive online help. Click Help on any Oracle Enterprise Manager page to display the online Help system.

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>Convention</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><code>monospace</code></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
This chapter provides information you need to build a ticketing connector and integrate it with Enterprise Manager. This chapter has following sections:

- **Introduction**
- **How a Ticketing Connector Functions**
- **Prerequisites**
- **Extracting Schema Files**
- **Building a Ticketing Connector**
- **Packaging and Deploying the Ticketing Connector**

---

**Note:** A ticketing connector is sometimes known as a help desk connector.

### 1.1 Introduction

Enterprise Manager Cloud Control 12c provides a Management Connector Framework (referred to as Connector Framework) to allow developers to build ticketing connectors that can be used to create/update tickets/incidents in an external ticketing application. Conceptually, a ticket and an incident are identical. To avoid confusion, *incident* will be used to refer to an incident in Enterprise Manager, and *ticket* will be used to refer to an incident in the external ticketing application.

For the connector to exchange data with the external ticketing application, a web service must be available that the connector can invoke to retrieve, create, and update incident information. The connector is composed of a set of XML and XSLT metadata files that define how the connector appears in the UI and how it connects to and exchanges data with the external ticketing application.

Ticketing connectors can be invoked in two different ways:

- **Auto Ticketing**
  
  Lets you configure the connector to automatically open or update a ticket whenever an incident or event is triggered in Enterprise Manager. You can specify incident rules for tickets to be created or updated.

- **Manual Ticketing**
  
  Lets you manually create a ticket from the Enterprise Manager console based on an open incident or event in Enterprise Manager. The connector populates the ticket with details based on the incident and the ticket template.
To use these ticketing features for your own help desk or ticketing system, you need to provide a set of metadata files. Table 1–1 lists the categories of metadata files that comprise a ticketing connector:

<table>
<thead>
<tr>
<th>Table 1–1 Metadata File Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
</tr>
<tr>
<td>Connector Descriptor</td>
</tr>
<tr>
<td>Ticket Request Templates</td>
</tr>
<tr>
<td>Response Templates</td>
</tr>
<tr>
<td>Publish Template</td>
</tr>
</tbody>
</table>

### 1.2 How a Ticketing Connector Functions

In order to know how to build a ticketing connector, you need to understand how one functions. The `connectorDeploy.xml` file, commonly referred to as the connector descriptor file, defines the connector in Enterprise Manager. When the connector is installed, the contents of the connector descriptor file are examined by Enterprise Manager to determine what fields are required by the connector and what templates to use when exchanging XML messages with the web service.

### Configuring the Connector

When the operator configures the connector, the configuration page is generated based on the information in the connector descriptor file. Typically, this includes fields that contain the URL and credentials to use when connecting to the web service. One of the fields that is required for a ticketing connector is the Ticket ID field. This field is used to verify connectivity to the external ticketing application and must be set to the identifier of an existing incident in the external ticketing application. When the operator clicks the **Ok** button on the configuration page, the Connector Framework performs the getTicket operation to retrieve information for the ticket identifier specified in the Ticket ID field. The Connector Framework uses the getTicket request template defined in the connector to generate the XML that is sent to the web service to retrieve the ticket information. When it gets the response from the web service, it uses the getTicket response template to translate the XML from the web service to the format expected by the Connector Framework. If the ticket data is successfully retrieved through the web service, the connector is marked as completed and is ready to use. If an error occurs, the configuration changes will be saved but the connector will not be marked as completed. Someone must investigate why it failed and address the problem. Once the problem has been addressed, they can go back to the configuration page and click **Ok** to have it test connectivity again.

### Creating a Ticket

Once the connector is marked as completed, you can manually create a ticket in the external ticketing application using an existing incident or you can set up incident rules to automatically create a ticket in the external ticketing application whenever
certain events occur. For example, you could set up a rule to open a ticket if any of your database tablespaces exceed a specified threshold. Whenever the rule is set up, you specify what target(s) the rule applies to, what criteria to use to trigger the rule, the ticketing connector to invoke and what template to use for that connector.

Whenever the criteria are met and the rule fires, the Connector Framework invokes the ticketing connector to create a ticket. To create the ticket:

1. The Connector Framework uses the template specified in the incident rule to translate the Enterprise Manager incident data format into an XML request that it sends to the web service.
2. The web service creates a ticket in the external ticketing application and sends a response with the identifier of the new ticket.
3. The Connector Framework uses the `createTicket` response template to translate the new incident identifier into the format expected by the Connector Framework.
4. When the Connector Framework gets the response, it saves the external ticket identifier in the incident TicketID field.

**Note:** The same template is used for create and update requests. The template uses the TicketID field to distinguish between a create and update. If the field has data, it knows it is an `updateTicket` operation being performed, otherwise it knows that it is a `createTicket` operation.

**Updating a Ticket**

Whenever something occurs that causes the incident in Enterprise Manager to be updated, the Connector Framework uses the template specified in the incident rule to translate the Enterprise Manager incident data format into an XML update request that it sends to the web service. The web service updates the ticket in the external ticketing application and sends a response. When the Connector Framework gets the response, it uses the `createTicket` response template to translate the response into a format expected by the Connector Framework. This response is used by the Connector Framework to verify that the update was completed successfully.

**Note:** There is no `updateTicket` response template. Create and update responses are handled by the template that is defined for the `createTicket` service operation.

**Clearing a Ticket**

When the incident is resolved and it goes into a cleared status, the Connector Framework performs a normal `updateTicket` operation with the `SeverityCode` field set to CLEAR.

**Synchronizing Ticket Status (Optional)**

There is also an optional feature that can be used to synchronize ticket status changes that occur in the external ticketing application. The feature requires that the external ticketing application be configured to call `emcli` whenever the status changes for a ticket that originated in Enterprise Manager. The call to `emcli` sends the new status to Enterprise Manager, and Enterprise Manager handles the request as a `publishTicket` operation. To make this work, a `publishTicket` request template must be defined to translate the data from `emcli` into the format expected by Enterprise Manager.
1.3 Prerequisites

You must have a good understanding of the XML, XSD, and XSLT technologies because you will be required to generate several XML and XSLT files during the course of building a connector. It is highly recommended that you familiarize yourself with these technologies before attempting to build a connector.

1.4 Extracting Schema Files

To create the ticketing and event connectors, you will need access to the schema files that define the format of the different files. The schema files are located in the Extensibility Development Kit (EDK). To install the EDK, go to the Setup menu, select Extensibility, then Development Kit. This page gives instructions for downloading and installing the EDK. Review the Requirements section and verify the prerequisites have been met before attempting to install the EDK. Once the prerequisites are confirmed, install the EDK as directed in the Deployment section.

The schema files are located in the emMrsXsds.jar file in the emSDK directory. To access the files, you will need to extract them using the jar command or any other utility that understands the jar file format. Use the following command to extract the files using the jar command from the EDK installation directory:

```
$JAVA_HOME/bin/jar xvf emSDK/emMrsXsds.jar
```

Table 1–2 shows the location of the extracted schema files. This table will be referenced in the different sections where the schema files are discussed.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectorDeploy.xsd</td>
<td>oracle/sysman/emSDK/core/connector/common</td>
</tr>
<tr>
<td>createTicket_response.xsd</td>
<td>oracle/sysman/emSDK/core/connector/ticketingConnector</td>
</tr>
<tr>
<td>EMEvent.xsd</td>
<td>oracle/sysman/emSDK/core/connector/eventConnector</td>
</tr>
<tr>
<td>EMIncident.xsd</td>
<td>oracle/sysman/emSDK/core/connector/ticketingConnector</td>
</tr>
<tr>
<td>getTicket_response.xsd</td>
<td>oracle/sysman/emSDK/core/connector/ticketingConnector</td>
</tr>
<tr>
<td>publishTicket.xsd</td>
<td>oracle/sysman/emSDK/core/connector/ticketingConnector</td>
</tr>
<tr>
<td>SelfUpdateManifest.xsd</td>
<td>oracle/sysman/emSDK/core/selfupdate/model</td>
</tr>
</tbody>
</table>

1.5 Building a Ticketing Connector

Now that you understand how a connector functions, you are now ready to start the process of building your ticketing connector. To build your connector, you will need to follow the instructions specified in the sections listed below:

- Determining Connector Functionality
- Developing Required Template Files
- Defining the Connector Descriptor File

1.5.1 Determining Connector Functionality

Before you can build a connector, you need to analyze your requirements and determine what functionality to include in the connector. This section assists you in determining what templates you will require for your ticketing connector. When you are done with this section, you should have a list of the templates that need to be implemented for your connector.
There are some templates that are required and must be included in every ticketing connector. You have no choice but to include these templates in your connector. There are other templates that are optional that may be included in the connector if deemed necessary. Table 1–3 lists the possible templates that can be defined for a ticketing connector. The Description column in this table explains the functionality provided by the template. You will need to analyze the functionality provided by the optional templates and determine which ones you want to include in your connector. Once you complete this analysis, you should have a list of templates that you need to provide. Your list will be comprised of the required templates plus the optional templates that you have selected for inclusion.

**Table 1–3 Possible Ticketing Templates**

<table>
<thead>
<tr>
<th>Template</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getTicket request</td>
<td>Required</td>
<td>Used to generate a request to retrieve ticket information from the web service.</td>
</tr>
<tr>
<td>getTicket response</td>
<td>Required</td>
<td>Used to translate the response from the web service to a format expected by Enterprise Manager.</td>
</tr>
<tr>
<td>Default template</td>
<td>Required</td>
<td>Used to generate a request to create/update a ticket that is sent to the web service. There must be at least one default template defined.</td>
</tr>
<tr>
<td>Additional Default templates</td>
<td>Optional</td>
<td>You can define additional templates that the operator can choose from when invoking the connector. Each default template has different functionality. For example one template may automatically close the ticket when the incident clears in Enterprise Manager. Another template will not close the ticket but update the history log.</td>
</tr>
<tr>
<td>createTicket response</td>
<td>Optional</td>
<td>Implement this template if you want to create a ticket and update it as incident updates occur in Enterprise Manager. Without this template, the identifier of the created ticket will not be saved in the Enterprise Manager incident. This will cause it to create a new ticket every time an update occurs. Although it is not required, it is highly recommended that you implement this template.</td>
</tr>
<tr>
<td>publishTicket request</td>
<td>Optional</td>
<td>This template is used to pick up ticket status changes that occur in the external ticketing application. When a ticket status change occurs, the external ticketing application must be configured to call emcli to send the status change to Enterprise Manager. This template handles the request and transforms it to the format expected by Enterprise Manager.</td>
</tr>
</tbody>
</table>

You will also need to determine the file name that you want to use for each template. There are no requirements on the template file names, but Oracle recommends that you use the following naming convention:

- `<methodName>_request.xml`
- `<methodName>_request.xsl`
- `<methodName>_response.xsl`

Table 1–4 lists the recommended filenames for the different templates based on the suggested naming convention.

**Table 1–4 Recommended Template Filenames**

<table>
<thead>
<tr>
<th>Template</th>
<th>Recommended Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>getTicket request</td>
<td>getTicket_request.xml</td>
</tr>
<tr>
<td>getTicket response</td>
<td>getTicket_response.xsl</td>
</tr>
<tr>
<td>Default templates</td>
<td>Choose a filename that describes the functionality of the template</td>
</tr>
<tr>
<td>createTicket response</td>
<td>createTicket_response.xsl</td>
</tr>
</tbody>
</table>
1.5.2 Developing Required Template Files

Now that you have identified the templates that you need to provide, the next step is to create the template files. It is highly recommended that you use XML/XSLT tools to generate and test the template files. You can create the files using a standard text editor but it will make the process much more difficult and time consuming. The tools catch format errors and allow you to test the templates before you package and install the connector in Enterprise Manager. This greatly reduces the number of corrections that you have to make to the installed connector. Each correction that you have to make to the connector requires that you uninstall the old connector, repackage and reinstall the new version of the connector.

The following subsections cover the steps required to create the different template files. You can ignore the sections that cover templates that are not targeted for your connector.

- `getTicket` Request Template
- `getTicket` Response Template
- Default Template(s)
- `createTicket` Response Template
- `publishTicket` Request Template

1.5.2.1 `getTicket` Request Template

The `getTicket` request template is a XML file that identifies the XML format required to retrieve the ticket information from the web service. If you have a sample of the XML required by the web service to retrieve the ticket information, you can copy that into the template file. If you do not have a sample, you will need to look at the WSDL or the schema that defines the format required by the web service. If you are not able to determine the XML format by manually looking at the files, there are tools that you can use that examine the WSDL/schema and generate a sample XML file.

There is at least one change that you need to make to the XML file to make it work as a template. In the location where the ticket identifier goes, you will need to replace whatever is specified with `@TicketId@`. This tells the Connector Framework to substitute the ticket identifier that is entered when the connector is configured.

Example 1–1 below is a sample of an XML file that was generated from a WSDL, and Example 1–2 shows the contents of the corresponding `getTicket` request template.

Since the sample included the SOAP envelope information, it was stripped from the template file. The SOAP information will be added by the Connector Framework whenever the web service is called. One other thing to note is that the definition for the `urn` namespace was moved because it was defined in the SOAP envelope that was deleted. If the `urn` namespace would not have been moved, it would have been an invalid request because the namespace would have been undefined. There is not really any testing that can be done on this template other than to use a tool to verify that the XML format is valid. If you can validate it against the schema that would be even better.

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```
Building a Ticketing Connector

Example 1–2  getTicket_request.xml

```xml
<urn:HelpDesk_Query_Service xmlns:urn="urn:HPD_IncidentInterface_Custom_WS">
  <urn:Incident_Number>@TicketId@</urn:Incident_Number>
</urn:HelpDesk_Query_Service>
```

1.5.2.2 getTicket Response Template

The getTicket response template is an XSLT file that is used to transform the response from the web service into the format expected by Enterprise Manager. The format of the response XML from the web service should be defined by the WSDL or by a schema. The format expected by Enterprise Manager is specified in the getTicket_response.xsd schema file. See Table 1–2 in the Extracting Schema Files section for the location of the getTicket_response.xsd schema file.

Example 1–3 shows a sample response file from the web service. Example 1–4 shows a sample XSLT template that is designed to transform the data to the format expected by Enterprise Manager. The XSLT template looks for a root element with a name of HelpDesk_Query_ServiceResponse that has a namespace of urn:HPD_IncidentInterface_Custom_WS. It creates a getTicketResponse root element with a namespace of http://xmlns.oracle.com/sysman/connector/tt and creates a child TicketId element and sets it to the identifier specified in the Incident_Number element. If the TicketId element is not empty, Enterprise Manager knows that the ticket was successfully retrieved. Example 1–5 shows the XML that was generated by performing the translation using an XSL translation tool.

Example 1–3  Input Response XML from Web Service

```xml
<?xml version='1.0' encoding='UTF-8'?>
<urn:HelpDesk_Query_ServiceResponse xmlns:urn="urn:HPD_IncidentInterface_Custom_WS">
  <urn:Incident_Number>TKT00001</urn:Incident_Number>
</urn:HelpDesk_Query_ServiceResponse>
```

Example 1–4  Sample XSLT Template File

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:urn="urn:HPD_IncidentInterface_Custom_WS"
  xmlns="http://xmlns.oracle.com/sysman/connector/tt"
  targetNamespace="http://xmlns.oracle.com/sysman/connector/tt"
  elementFormDefault="qualified">
  <xsl:template match="urn:HelpDesk_Query_ServiceResponse">
    <getTicketResponse>
      <TicketId><xsl:value-of select="urn:Incident_Number/text()"/></TicketId>
    </getTicketResponse>
  </xsl:template>
</xsl:transform>
```
Example 1–5  File Resulting from Transformation

```xml
<?xml version="1.0"?>
<getTicketResponse xmlns="http://xmlns.oracle.com/sysman/connector/tt"
   xmlns:urn="urn:HPD_IncidentInterface_Custom_WS">
  <TicketId>TKT00001</TicketId>
</getTicketResponse>
```

Whenever you create your XSLT template, you do not have to start from scratch. It is easier to copy an existing template and customize it to fit your situation. To customize the template shown above:

1. Replace the `urn` namespace definition with the namespace that will be specified in the XML coming from your web service.
2. Change the template match attribute to reference the root element in the XML coming from your web service.
3. Change the name of the element where it gets the ticket identifier information when it creates the `TicketId` element.
4. Once you create your file, you should run an XSLT tool to test your template by transforming data from a sample XML and verify that it generates the correct XML format.

1.5.2.3 Default Template(s)

The default templates are the most difficult to build because they are larger and more complicated than the other templates. When a ticket is manually created or an incident rule fires that invokes the ticketing connector, the Connector Framework generates an internal XML document that contains data for the affected incident. This generated XML document is used as the input XML for a translation involving the selected default template. The XML that results from the translation is the XML that will be sent to the web service to create or update a ticket in the external ticketing application.

Before you can build your template, you need to understand the format of the data being created/updated in the external ticketing application and the format of the data coming from Enterprise Manager. Understanding the format involves identifying the fields that are available and the values that can be entered in those fields.

You need to familiarize yourself with the format of the data that is specified to create/update a ticket in the external ticketing system. A good place to start is the WSDL or a schema file that defines the format of the data. You will also need to see sample create/update requests to see how the data is formatted. If samples are not available, you should be able to manually retrieve data for an existing ticket using a XML client tool. This should give you a good idea of what the data looks like.

Once you understand the data in the external ticketing application, you then need to study and understand the data coming from Enterprise Manager. The fields that are available in the Enterprise Manager incident/event data are identified in two schema files. The `EMIncident.xsd` file defines the format of the incident data and the `EMEvent.xsd` file defines the format of the event data that triggered the incident. See Table 1–2 in the Extracting Schema Files section for the location of the `EMIncident.xsd` and `EMEvent.xsd` schema files. The schema files identify the fields and tell the type of data in those fields but do not give a good indication of what data is actually present. To get a good idea of what the data looks like, you need a sample XML file that was generated by Enterprise Manager. Appendix F, "Sample Incident Data," shows sample transactions that were generated by Enterprise Manager.

Once you are familiar with the data on both ends, you need to determine how many templates you need and the mapping that will be performed by each. This involves
determining what fields you will specify in create/update requests and the format of the data for those fields. You only have two choices on the source of the data. You can hard code the data or get it from the Enterprise Manager incident/event data. You can make the mapping as sophisticated or simple as you like. Ultimately, you will need to determine what fields and settings make sense for your environment.

Once you have identified the templates and the mapping of each, you are now ready to build the XSLT file(s). To start the first template, copy the XML shown in Example 1–6 into your editor where you are building the XSLT file. This contains the basic skeleton that you will need to build your template. The skeleton has a section for create requests and another section for update requests. You will need to add your mapping logic in the appropriate sections. If multiple templates are being built, the recommended approach is to build the simplest template first and to test it thoroughly. Once you have verified that it works, you can make a copy of the template and use that as a baseline for the other templates.

Appendix D, "Default Template Example," walks through an example that shows how to build a default template.

**Example 1–6 Template Skeleton**

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version="1.0"

xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
xmlns:emcf='http://xmlns.oracle.com/sysman/connector'>

<xsl:template match='emcf:EMIncident'>
<xsl:choose>
<xsl:when test="normalize-space(emcf:TicketID) = ''">
<!-- CREATE Request -->
<!-- Replace this comment with the mapping logic for create requests -->
</xsl:when>
<xsl:otherwise>
<!-- UPDATE Request -->
<!-- Replace this comment with the mapping logic for update requests -->
</xsl:otherwise>
</xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

**1.5.2.4 createTicket Response Template**

The `createTicket` response template is an XSLT file that is used to transform the response from the web service into the format expected by Enterprise Manager. Although the template name implies it is only for create responses, it is also used to handle update responses. The format of the response XML from the web service should be defined by the WSDL or by a schema. The format expected by Enterprise Manager is specified in the `createTicket_response.xsd` schema file. See Table 1–2 in the Extracting Schema Files section for the location of the `createTicket_response.xsd` schema file.

**Example 1–7** shows a sample response file from the web service. **Example 1–8** shows a sample XSLT template that is designed to transform the data to the format expected by Enterprise Manager. The XSLT template looks for a root element with a name of `HelpDesk_Submit_ServiceResponse` that has a namespace of `urn:HPD_IncidentInterface_Create_WS`. It creates a `CreateTicketResponse` root element with a namespace of `http://xmlns.oracle.com/sysman/connector` and creates a child `TicketId` element and sets it to the identifier specified in the `Incident_Number` element. It also populates the `Incident_Number` variable with the identifier specified in
the Incident_Number element. If the TicketId element is not empty, Enterprise Manager knows that the ticket was successfully created. Example 1–9 shows the XML that was generated by performing the translation using an XSLT tool.

**Example 1–7  Input Response XML from Web Service**

```xml
<?xml version='1.0' encoding='UTF-8'?>
<urn:HelpDesk_Submit_ServiceResponse xmlns:urn="urn:HPD_IncidentInterface_Create_WS">
  <urn:Incident_Number>TKT00001</urn:Incident_Number>
</urn:HelpDesk_Submit_ServiceResponse>
```

**Example 1–8  Sample XSLT Template File**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:urn="urn:HPD_IncidentInterface_Create_WS" xmlns=http://xmlns.oracle.com/sysman/connector">
  <xsl:template match="urn:HelpDesk_Submit_ServiceResponse">
    <CreateTicketResponse>
      <TicketId>
        <xsl:value-of select="urn:Incident_Number"/>
      </TicketId>
      <InstanceVariable>
        <VariableName>Incident_Number</VariableName>
        <VariableValue>
          <xsl:value-of select="urn:Incident_Number"/>
        </VariableValue>
      </InstanceVariable>
    </CreateTicketResponse>
  </xsl:template>
</xsl:transform>
```

**Example 1–9  File Resulting from Transformation**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CreateTicketResponse xmlns="http://xmlns.oracle.com/sysman/connector" xmlns:urn="urn:HPD_IncidentInterface_Create_WS">
  <TicketId>TKT00001</TicketId>
  <InstanceVariable>
    <VariableName>Incident_Number</VariableName>
    <VariableValue>TKT00001</VariableValue>
  </InstanceVariable>
</CreateTicketResponse>
```

Whenever you create your XSLT template, it is easier to copy an existing template and customize it to fit your situation. To customize the template shown above:

1. Replace the urn namespace definition with the namespace that will be specified in the XML coming from your web service.
2. Change the template match attribute to reference the root element in the XML coming from your web service.
3. Change the name of the element where it gets the ticket identifier information when it creates the TicketId element.
4. Once you create your file, you should run an XSLT tool to test your template by transforming data from a sample XML and verify that it generates the correct XML format.
1.5.2.5 **publishTicket Request Template**

The `publishTicket` request template is an XSLT file that is used to transform status change information sent by emcli into the format expected by Enterprise Manager. The format expected by Enterprise Manager is specified in the `publishTicket.xsd` schema file. See Table 1–2 in the Extracting Schema Files section for the location of the `publishTicket.xsd` schema file.

Since the source (emcli) and the destination (Enterprise Manager) XML formats will be the same for all connectors, the template defined will require little or no changes from one connector to another. The only part that may require changes is the status field. Some applications have two status values defined for each possible status. One is used internally, and the other is the value that is displayed at the console. For example, an application may have numeric internal status values that are displayed as strings at the console. If values that emcli passes for status are the display strings, you can pass the status straight through to Enterprise Manager without modification. If the values are passed as internal values, you will need to translate the numeric value to the corresponding string value.

As an example, assume there is an external application that tracks status internally using a numeric value to represent the status. Also assume that at the console, the status is displayed as a text string and that the internal status of 2 is displayed as "In Progress" at the console. If emcli passes the internal status value of 2, the template will need to translate the 2 to a value of "In Progress" and set the status field to "In Progress". If emcli passes the display status of "In Progress", the value can be passed through without any modifications. Example 1–10 contains a template that passes the status through without modifications, and Example 1–11 contains a template that translates the status. You will want to copy the template that applies to your situation. If no translation is required, the template in Example 1–10 can be copied and used without any modifications. If translation is required, the template in Example 1–11 can be copied and the status translation mapping will need to be modified to use the values defined in your application.

### Example 1–10  Publish Template that Passes Unmodified Status

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version='1.0'
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
xmns:a='http://xmlns.oracle.com/sysman/connector'
targetNamespace='http://xmlns.oracle.com/sysman/connector'
elementFormDefault='qualified'>

<xsl:template match='a:InboundData'>
  <InboundData>
    <Operation><xsl:value-of select="a:Operation"/></Operation>
    <PropertyList>
      <ticket_guid><xsl:value-of select="a:ticket_guid"/></ticket_guid>
      <status><xsl:value-of select="a:status/text()"/></status>
      <connector_guid><xsl:value-of select="a:connector_guid"/></connector_guid>
      <last_updated_date><xsl:value-of select="a:last_updated_date"/></last_updated_date>
    </PropertyList>
  </InboundData>
</xsl:template>

</xsl:transform>
```

### Example 1–11  Publish Template that Passes Translated Status

```xml
<?xml version='1.0' encoding='UTF-8'?>
<InboundData>
  <Operation>
    <ticket_guid><xsl:value-of select="a:ticket_guid"/></ticket_guid>
    <status><xsl:value-of select="a:status/text()"/></status>
    <connector_guid><xsl:value-of select="a:connector_guid"/></connector_guid>
    <last_updated_date><xsl:value-of select="a:last_updated_date"/></last_updated_date>
  </Operation>
</InboundData>
```
1.5.3 Defining the Connector Descriptor File

Now that you have the templates created, it is time to create a connector descriptor file that defines the connector in Enterprise Manager. The connector descriptor XML file describes the connector metadata and the configuration properties of the connector, such as web service end points and authentication schema.

The key points to remember when constructing a descriptor are

- The connector descriptor file name must be `connectorDeploy.xml`
- The XML file should adhere to the `connectorDeploy.xsd` schema file.

See Table 1–2 in the Extracting Schema Files section for a summary for the location of schema files.

Refer to the sample `connectorDeploy.xml` in Appendix A, "Ticketing Connector Samples" for reference implementation.

Table 1–5 lists the sections that comprise a connector descriptor and provides a summary of what each section does:

<table>
<thead>
<tr>
<th>Metadata Section</th>
<th>Required</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector Information</td>
<td>Yes</td>
<td>This section provides information about the connector that will be displayed at the UI.</td>
</tr>
<tr>
<td>Authentication</td>
<td>No</td>
<td>Specifies the authentication method and the parameters that are required to connect to the web service.</td>
</tr>
<tr>
<td>Connectivity Test Variable</td>
<td>Yes</td>
<td>The variable defined in this section is used during the initial configuration step to validate connectivity.</td>
</tr>
</tbody>
</table>
The following sections provide detailed information about the contents of the connector descriptor:

- Connector Deploy Field Information
- Connector Information Section
- Sample Connector Information Section
- Authentication Section
- Sample Authentication Section
- Connectivity Test Variable Section
- Sample Connectivity Test Variable Section
- External URL Section
- Sample External URL Section
- Service Section
- Sample Service Section
- Template Registration Section
- Sample Template Registration Section
- Complete Connector Deployment File

### 1.5.3.1 Connector Deploy Field Information

Each section of the connector deployment file contains fields that provide specific information about the connector. The following sections contain detailed information about what fields are available and what data goes in those fields.

### 1.5.3.2 Connector Information Section

The Connector Information section provides information about the connector such as name, version, description, etc. that will be displayed at the UI. Table 1–6 lists the fields in this section and provides an explanation of each field.

<table>
<thead>
<tr>
<th>Metadata Section</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>External URL</td>
<td>No</td>
<td>This section enables you to configure the connector to provide a link to directly view the ticket contents in the external system.</td>
</tr>
<tr>
<td>Service</td>
<td>Yes</td>
<td>This section is used to configure the URLs used to connect to the web service for the different service operations.</td>
</tr>
<tr>
<td>Template Registration</td>
<td>Yes</td>
<td>This section is used to define all of the templates that are used for the connector.</td>
</tr>
</tbody>
</table>

The following sections provide detailed information about the contents of the connector descriptor:

- Connector Deploy Field Information
- Connector Information Section
- Sample Connector Information Section
- Authentication Section
- Sample Authentication Section
- Connectivity Test Variable Section
- Sample Connectivity Test Variable Section
- External URL Section
- Sample External URL Section
- Service Section
- Sample Service Section
- Template Registration Section
- Sample Template Registration Section
- Complete Connector Deployment File

### 1.5.3.1 Connector Deploy Field Information

Each section of the connector deployment file contains fields that provide specific information about the connector. The following sections contain detailed information about what fields are available and what data goes in those fields.

### 1.5.3.2 Connector Information Section

The Connector Information section provides information about the connector such as name, version, description, etc. that will be displayed at the UI. Table 1–6 lists the fields in this section and provides an explanation of each field.

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The connector name that will be displayed at the UI.</td>
</tr>
<tr>
<td>Version</td>
<td>Yes</td>
<td>The connector version that will be displayed at the UI.</td>
</tr>
<tr>
<td>EMCompatibleVersion</td>
<td>Yes</td>
<td>The earliest version of Enterprise Manager that is supported.</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>The connector description that will be displayed at the UI.</td>
</tr>
</tbody>
</table>
1.5.3.3 Sample Connector Information Section

Example 1–12 shows a sample of the information that is included in the Connector Information section. All of the fields in this section are contained in the ManagementConnector node.

Example 1–12 Connector Information Section Sample

<Name>Remedy Service Desk 7.6 Connector</Name>
<Version>12.1.0.2.0</Version>
<EMCompatibleVersion>12.1.0.1.0</EMCompatibleVersion>
<Description>Remedy Service Desk 7.6.04 Integration with Enterprise Manager</Description>
<Category>TicketingConnector</Category>

1.5.3.4 Authentication Section

The authentication section specifies the authentication method and the credentials that are required to connect to the web service. There are three possible authentication types that can be configured. If no authentication section is specified, no authentication will be performed when the connector connects to the web service. Table 1–7 lists the three possible authentication sections and the fields contained in each.

Table 1–7 Authentication Fields

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAPHeaderAuthentication</td>
<td>No</td>
<td>Specifies the credentials used to connect to the web service using SOAP header authentication.</td>
</tr>
<tr>
<td>*Username</td>
<td>Yes</td>
<td>The username to specify in the SOAP header</td>
</tr>
<tr>
<td>*Password</td>
<td>Yes</td>
<td>The password to specify in the SOAP header</td>
</tr>
<tr>
<td>*AuthVariable</td>
<td>No</td>
<td>Up to 20 other variables to pass in the SOAP header</td>
</tr>
<tr>
<td>*SOAPHeader</td>
<td>Yes</td>
<td>A string that serves as template for the SOAP header. It will be updated by substituting the user inputs for variables defined above in the designated location and bound with a HTTP request.</td>
</tr>
<tr>
<td>HTTPBasicAuthentication</td>
<td>No</td>
<td>Specifies the credentials used to connect to the web service using basic authentication</td>
</tr>
<tr>
<td>*Username</td>
<td>Yes</td>
<td>The username to specify when calling the web service</td>
</tr>
<tr>
<td>*Password</td>
<td>Yes</td>
<td>The password to specify when calling the web service</td>
</tr>
<tr>
<td>UserNameTokenAuthentication</td>
<td>No</td>
<td>Specifies the credentials used to connect to the web service using Username Token Profile authentication</td>
</tr>
<tr>
<td>*Username</td>
<td>Yes</td>
<td>The username to specify</td>
</tr>
<tr>
<td>*Password</td>
<td>Yes</td>
<td>The password to specify</td>
</tr>
</tbody>
</table>
* Fields marked with an asterisk are comprised of the following subfields:
  - **VariableName**: Name of the variable being defined
  - **DisplayName**: Name to use when displaying information about this field at the UI
  - “required” attribute: Specifies whether the field is required (defaults to `false` if not specified)

**1.5.3.5 Sample Authentication Section**

Example 1–13 shows the information that is included in the Authentication section. In this example, the authentication method is a SOAP header. The operator would be required to provide Remedy Username, Remedy Password, Authentication, Locale, and Timezone values when configuring the connector. The values entered by the operator would be used to populate the XML in the SOAPHeader section, and this would be passed in the SOAP header for any requests that are sent to the web service.

**Example 1–13 Authentication Section Sample**

```xml
<SOAPHeaderAuthentication>
  <Username required="true">
    <VariableName>USERNAME</VariableName>
    <DisplayName>Remedy Username</DisplayName>
  </Username>
  <Password>
    <VariableName>PASSWORD</VariableName>
    <DisplayName>Remedy Password</DisplayName>
  </Password>
  <AuthVariable>
    <VariableName>AUTHENTICATION</VariableName>
    <DisplayName>Authentication</DisplayName>
  </AuthVariable>
  <AuthVariable>
    <VariableName>LOCALE</VariableName>
    <DisplayName>Locale</DisplayName>
  </AuthVariable>
  <AuthVariable>
    <VariableName>TIMEZONE</VariableName>
    <DisplayName>Timezone</DisplayName>
  </AuthVariable>
</SOAPHeader>

<![CDATA[
<urn:AuthenticationInfo xmlns:urn="urn:HelpDesk_Submit_Service">
<urn:userName>$USERNAME$</urn:userName>
<urn:password>$PASSWORD$</urn:password>
<urn:authentication>$AUTHENTICATION$</urn:authentication>
<urn:locale>$LOCALE$</urn:locale>
<urn:timeZone>$TIMEZONE$</urn:timeZone>
</urn:AuthenticationInfo>
]]></SOAPHeader>
</SOAPHeaderAuthentication>

**1.5.3.6 Connectivity Test Variable Section**

This section defines the variable that is used by the getTicket service operation during the initial configuration step to validate connectivity with the web service. Table 1–8 lists the fields in the section and provides an explanation of each field.
1.5.3.7 Sample Connectivity Test Variable Section

Example 1–14 shows the information that is included in the Connectivity Test Variable section. You should be able to use this section without any modifications.

Example 1–14 Connectivity Test Variable Section Sample

```<ConnectivityTestVariable>
  <VariableName>TICKET_ID</VariableName>
  <DisplayName>Ticket ID</DisplayName>
</ConnectivityTestVariable>`

1.5.3.8 External URL Section

This section enables you to configure the connector to provide a link to directly view the ticket contents in the external system. The ticketing application must provide a link that can be used to directly access the ticket information. Table 1–9 lists the fields in the section and provides an explanation of each field.

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalURL</td>
<td>No</td>
<td>URL that can be used to access the ticket in the external system.</td>
</tr>
<tr>
<td>Pattern</td>
<td>Yes</td>
<td>Pattern used to determine the external URL. The value of the <code>&lt;Pattern&gt;</code> tag describes the URL string, and how user-configured variables are inserted into it.</td>
</tr>
<tr>
<td>ConfigVariable</td>
<td>No</td>
<td>Variables that must be specified by the operator when the connector is configured. The values that the user provides for each user variable is inserted into the URL pattern string accordingly. If there is a user variable &quot;X&quot; then the user input value replaces &quot;$X$&quot; when the ticket URL is generated. There can be up to 50 variables specified.</td>
</tr>
<tr>
<td>VariableName</td>
<td>Yes</td>
<td>Name of the variable being defined.</td>
</tr>
<tr>
<td>DisplayName</td>
<td>Yes</td>
<td>Name to use when displaying information about this field at the UI.</td>
</tr>
<tr>
<td>required attribute</td>
<td>No</td>
<td>Specifies whether the field is required - defaults to <code>false</code> if not specified.</td>
</tr>
</tbody>
</table>
1.5.3.9 Sample External URL Section

Example 1–15 shows the information that is included in the External URL section. The variables defined in the ConfigVariable section will be displayed as input fields on the configuration page, and the operator will enter values in the fields. The values that they enter will be plugged into the URL where the variable name is listed (surrounded by $) in the Pattern field. Also the @Incident_Number@ will be replaced by the contents of the Incident_Number variable that was created by the createTicket response template shown in Example 1–9.

Example 1–15 External URL Section Sample

```
<ExternalURL>
  <Pattern>
    <![CDATA[http://$WEB_SERVER$/arsys/forms/$ARSERVER_NAME$/FORM_NAME/?qual=%27Incident%20Number*%27=%22@Incident_Number@%22]]>
  </Pattern>
  <ConfigVariable required="true">
    <VariableName>WEB_SERVER</VariableName>
    <DisplayName>Web Server</DisplayName>
  </ConfigVariable>
  <ConfigVariable required="true">
    <VariableName>FORM_NAME</VariableName>
    <DisplayName>HelpDesk Case Form Name</DisplayName>
  </ConfigVariable>
  <ConfigVariable required="true">
    <VariableName>ARSERVER_NAME</VariableName>
    <DisplayName>ARServer Name</DisplayName>
  </ConfigVariable>
</ExternalURL>
```

1.5.3.10 Service Section

This section is used to configure the URLs used to connect to the web service for the different service operations. There must be a separate Service section entry for each of the following operations:

- **createTicket**
  Create ticket on the external system.

- **updateTicket**
  Forward incident updates to the external system.

- **getTicket**
  Service method to validate the ticket connector connectivity with Enterprise Manager.

**Note:** The service names in the connector descriptor should exactly match the names defined above and are case sensitive.

Table 1–10 lists the fields in the section and provides an explanation of each field.

**Table 1–10 Service Fields**

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Yes</td>
<td>This section allows you to specify configurations specific to the Ticketing System’s Web services.</td>
</tr>
</tbody>
</table>
Example 1–16 shows all three required Service sections. The URLs in the WebServiceEndpoint element are placed in a CDATA section to avoid conflicts with reserved XML characters. The values surrounded by square brackets [] need to be replaced by the operator on the configuration page. For example, the default URL for the createTicket operation is

http://[midtier_server]/arsys/services/ARService?server=[servername]&webService=HPD_IncidentInterface_Create_WS

This value will be displayed in the Web Service End Points section of the configuration page next to the createTicket operation. The operator will need to replace [midtier_server] with the actual midtier server IP address or hostname. They would also need to replace [servername] with the server IP address or hostname.

Example 1–16 Service Section Sample

Table 1–10 (Cont.) Service Fields

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Yes</td>
<td>Method defines one of the EM-specific service operation names. For ticketing connectors, it must be set to one of the following values: getTicket createTicket updateTicket</td>
</tr>
<tr>
<td>WebServiceEndpoint</td>
<td>Yes</td>
<td>This field specifies the default Web service endpoint string to be displayed in the Web service section of the Management Connector page.</td>
</tr>
<tr>
<td>SOAPAction</td>
<td>No</td>
<td>The SOAPAction to specify when calling the web service</td>
</tr>
<tr>
<td>SOAPBindingType</td>
<td>No</td>
<td>Possible values are: SOAP11HTTP_BINDING SOAP11HTTP_BINDING MTOM_BINDING SOAP12HTTP_BINDING MTOM_BINDING</td>
</tr>
</tbody>
</table>

1.5.3.11 Sample Service Section
1.5.3.12 Template Registration Section

This section is used to define all of the templates that were built in the previous chapter. Table 1–11 lists the fields in the section and provides an explanation of each field.

<table>
<thead>
<tr>
<th>Table 1–11</th>
<th>Template Registration Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Section/Field</strong></td>
<td><strong>Required</strong></td>
</tr>
<tr>
<td>TemplateRegistration</td>
<td>Yes</td>
</tr>
<tr>
<td>FileName</td>
<td>Yes</td>
</tr>
<tr>
<td>InternalName</td>
<td>Yes</td>
</tr>
<tr>
<td>TemplateName</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Must not contain any restricted characters. Only the following characters are allowed:
  - Upper case alphabetic characters (A-Z)
  - Lower case alphabetic characters (a-z)
  - Numeric characters (0-9)
  - Underscore character (_)
- Must not be set to the following service method names:
  - createTicket
  - publishTicket
  - getTicket
1.5.3.13 Sample Template Registration Section

Example 1–17 shows the different TemplateRegistration sections.

Example 1–17 Template Registration Section Sample

"<TemplateRegistration>
  <FileName>getTicket_request.xml</FileName>
  <InternalName>getTicket</InternalName>
  <TemplateName>Get Ticket</TemplateName>
  <TemplateType>OutboundXML</TemplateType>
  <Description>This is the getTicket request template.</Description>
</TemplateRegistration>

<TemplateRegistration>
  <FileName>getTicket_response.xsl</FileName>
  <InternalName>getTicket</InternalName>
  <TemplateName>Get Ticket</TemplateName>
  <TemplateType>InboundXSL</TemplateType>
  <Description>This is the getTicket response template.</Description>
</TemplateRegistration>

<TemplateRegistration>
  <FileName>createTicket_response.xsl</FileName>
  <InternalName>createTicket</InternalName>
  <TemplateName>Create Ticket Response</TemplateName>
  <TemplateType>InboundXSL</TemplateType>
  <Description>This is the create ticket response template.</Description>
</TemplateRegistration>

<TemplateRegistration>
  <FileName>templates/Remedy_DefaultCategory AutoClose.xsl</FileName>
  <InternalName>Remedy_DefaultCategory AutoClose.xsl</InternalName>
  <TemplateName>Remedy Default Category Auto Close</TemplateName>
  <TemplateType>OutboundXSL</TemplateType>
  <Description>This is the Remedy default template with auto close function.</Description>
</TemplateRegistration>"
1.5.3.14 Complete Connector Deployment File

Example A–1 in Appendix A shows the complete connector deployment file that includes the samples shown in the preceding sections. Figure 1–1 shows an example of the connector configuration page that is displayed for the sample connector shown in Appendix A. The image has been labeled to show where the fields were defined in the deployment file.

Figure 1–1 Complete Connector Deployment Page

1.6 Packaging and Deploying the Ticketing Connector

Enterprise Manager uses the Self Update feature to deploy the connector. This feature, which can be accessed through the console, provides the ability to import the connector into the Enterprise Manager environment.

To deploy the connector:

1. Prepare the connector jar file.

   Package all XML and XSLT template files as a .jar file

   ```
   <name>_connector.jar
   --> connectorDeploy.xml
   ```
2. Prepare the manifest file.

The SelfUpdateManifest.xsd schema file defines the format of the manifest file.

See Table 1–2 in the Extracting Schema Files section for the location of the SelfUpdateManifest.xsd schema file.

Key attributes of the self update manifest files are:

- **EntityType**
  - Value is `core_connector`
- **EntityTypeVersion**
  - Current release version. Value=12.1.0.1.0
- **Version**
  - Version number of the connector. This value must be set to the value specified in the ManagementConnector/Version node in the connectorDeploy.xml file.
- **Attribute @Name=connector_type**
  - Connector type name. This value must be set to the value specified in the ManagementConnector/Name node in the connectorDeploy.xml file.
- **Attribute @Name=connector_category**
  - Category type can be TicketingConnector or EventConnector
- **ArchiveList**
  - This element contains the list of archives that are part of the connector setup. Generally there will be a single connector jar, but for some special implementations there may be additional jars (adapter or agent). In these cases, the connector specific jar should be the first one in the defined list. This is a mandatory requirement.

**Example 1–18 shows the code for the connector_manifest.xml file.**

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<EntityInstance
    xmlns="http://www.oracle.com/EnterpriseGridControl/SelfUpdateManifest"
    EntityTypeVersion="12.1.0.1.0"
    EntityType="core_connector"
    Maturity="PRODUCTION"
    Vendor="Oracle"
    PluginID="oracle.sysman.core"
>
    <Description><![CDATA[BMC Remedy 7.6.04 Service Desk Connector - 12.1.0.2.0]]></Description>
    <AttributeList>
        <Version>12.1.0.2.0</Version>
        <Attribute Name="connector_type" Value="Remedy Service Desk 7.6 Connector"
            Label="Remedy Service Desk 7.6 Connector" />
        <Attribute Name="connector_category" Value="TicketingConnector"
            Label="Ticketing Connector" />
    </AttributeList>
</EntityInstance>
```
Oracle Management Connector for Remedy Service Desk integrates Oracle Enterprise Manager Grid Control’s proactive alert detection and resolution features with BMC’s Remedy 7.6.04 Service Desk capabilities to provide a seamless workflow for incident management and resolution.

Remedy Service Desk 7.6 Connector works with BMC Remedy ITSM 7.6.04 Incident Management Application.

Change Logs:

12.1.0.2.0
- Initial Release

3. Configure emedk tool

The emedk tool can be configured by following instructions from Enterprise Manager. From the Setup menu, select Extensibility, then Development Kit.

4. Prepare the self-update archive

This requires the connector jar file and the manifest file for the connector. To prepare self-update, call the following utility to create a self update archive file:

```bash
edkutil prepare_update
    -manifest "manifest xml"
    -archivedir "archives directory"
    -out "output file or directory"
    [-typexml "update type xml"]
```

Table 1–12 Self Update Utility Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-manifest</td>
<td>Self update manifest file that describes the update.</td>
</tr>
<tr>
<td>-archivedir</td>
<td>Directory containing the archive files specified in the manifest file.</td>
</tr>
<tr>
<td>-out</td>
<td>Directory or filename for the self update archive. If a directory is specified, the filename is autogenerated.</td>
</tr>
<tr>
<td>-typexml</td>
<td>Optional path to the update type xml</td>
</tr>
</tbody>
</table>

The following example creates a self update archive in the /u01/sar directory based on the manifest file /u01/connector/connector_manifest.xml. The
archives referred to in `connector_manifest.xml` are picked from the directory `/u01/connector/archives`.

```
edkutil prepare_update
    -manifest /u01/connector/connector1_manifest.xml
    -archivedir /u01/connector/archives
    -out /u01/sar/sample_connector.zip
```

5. Import the connector archive to Enterprise Manager by calling any one of the following `emcli` commands:

```
emcli import_update -file=file\ -omslocal
```

```
emcli import_update
    -file=file\ -host=hostname\ [-credential_set_name=setname\ ] -credential_name=name -credential_owner=owner
```

These commands import a Self Update archive file into Enterprise Manager. On successful import, the update is displayed on the Self Update Home in downloaded status for further action.

*Table 1–13* describes the options available with this command:

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-file</code></td>
<td>The complete pathname of the update archive file</td>
</tr>
<tr>
<td><code>-omslocal</code></td>
<td>The flag specifying that the file is accessible from the OMS</td>
</tr>
<tr>
<td><code>-host</code></td>
<td>The target name for a host target where the file is available</td>
</tr>
<tr>
<td><code>-credential_set_name</code></td>
<td>The set name of the preferred credential stored in the repository for the host target. Can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- HostCredsNormal</td>
</tr>
<tr>
<td></td>
<td>Default unprivileged credential set</td>
</tr>
<tr>
<td></td>
<td>- HostCredsPriv</td>
</tr>
<tr>
<td></td>
<td>Privileged credential set</td>
</tr>
<tr>
<td><code>-credential_name</code></td>
<td>The name of a named credential stored in the repository. This option must be specified along with <code>-credential_owner</code> option.</td>
</tr>
<tr>
<td><code>-credential_owner</code></td>
<td>The owner of a named credential stored in the repository. This option must be specified along with <code>-credential_name</code> option.</td>
</tr>
</tbody>
</table>

The following paragraphs provide some examples of the use of the `emcli` command:

**Example 1**

Imports the file `sample_connector.zip`. The file must be present on the OMS host. In a multiple OMS setup, the request can be processed by any OMS, so the file should be accessible from the OMS processing the request. This usually means that the file must be kept on a shared location that is accessible from all OMS.

```
emcli import_update
    -file=/u01/sar/sample_connector.zip
    -omslocal
```

**Example 2**
Imports the file `sample_connector.zip.zip` that is present on the `host1.example.com` host. The host must be a managed host target in Enterprise Manager and the agent on this host must be up and running. The preferred unprivileged credentials for `host1.example.com` are used to retrieve the remote file.

```
emcli import_update
-file=/u01/sar/sample_connector.zip
-host=host1.example.com
-credential_set_name=HostCredsNormal
```

**Example 3**

Imports the file `sample_connector.zip` that is present on the `host1.example.com` host. The host must be a managed host target in Enterprise Manager and the agent on this host must be up and running. The named credentials `host1_creds` owned by user `admin1` are used to retrieve the remote file.

```
emcli import_update
-file=/u01/sar/sample_connector.zip
-host=host1.example.com
-credential_name=host1_creds
-credential_owner=admin1
```

6. Apply the connector using one of the following methods:

- From the Cloud Control console:
  a. Go to Self-Update Home page. The connector will be shown as downloaded.
  b. Select the connector row and click Apply to deploy the connector.

- From the command line:
  a. Run the following `emcli list` command to determine the identifier of the connector that was just imported:

    ```
    emcli list -resource=Updates -bind="et_name = 'core_connector'"
    ```

    The output of the command would look like this example:

    | Status   | Category    | Type              | Version   | Id                 |
    |----------|-------------|-------------------|-----------|--------------------|
    | Applied  | Ticketing   | Remedy Service    | 12.1.0.1.0| 123456789ABCDE     |
    |          | Connector   | Desk Connector    |           |                    |
    | Applied  | Event       | HP OMU Connector  | 12.1.0.3.0| 11223344AABBCC     |
    |          | Connector   | Desk 7.6 Connector|           |                    |
    | Applied  | Ticketing   | Remedy Service    | 12.1.0.3.0| 1A2B3C4D5E6F7G     |
    |          | Connector   | Desk 7.6 Connector|           |                    |
    | Applied  | Ticketing   | CASD Connector    | 12.1.0.3.0| 5443322CCBBAA      |
    |          | Connector   | Desk 7.6 Connector|           |                    |

    Note the ID for the connector that was just imported.

  b. Run the following `emcli apply_updates` command using the connector ID from the previous step:

    ```
    emcli apply_updates -id=<ID>
    ```
This chapter provides information that explains how to build an event connector and integrate it with Enterprise Manager.

This chapter has the following sections:

- **Introduction**
- **How an Event Connector Functions**
- **Prerequisites**
- **Extracting Schema Files**
- **Building an Event Connector**
- **Packaging and Deploying the Event Connector**

### 2.1 Introduction

Enterprise Manager Cloud Control 12c provides a Management Connector Framework (referred to as Connector Framework) to allow developers to build event connectors that can be used to create/update events in an external application.

For the connector to exchange data with the external application, a web service must be available that the connector can invoke to create and update event information. The connector is composed of a set of XML and XSLT metadata files that define how the connector appears in the UI and how it connects to and exchanges data with the external application.

To create an event connector for your own system, you need to provide a set of metadata files. Table 2–1 lists the categories of metadata files that comprise an event connector:

<table>
<thead>
<tr>
<th>Table 2–1 Metadata File Categories</th>
<th>Category</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Connector Descriptor</td>
<td>XML</td>
<td>The connector descriptor file defines the connector in Enterprise Manager. The file contains information about how the connector will appear in the UI, where the web service is located, how to connect to it, and what templates to use to translate data sent between systems.</td>
</tr>
<tr>
<td></td>
<td>Request Templates</td>
<td>XML or XSL</td>
<td>These templates are used to generate XML requests that are sent to the web service to create or update an event. They translate the Enterprise Manager event data fields into the format expected by the web service.</td>
</tr>
<tr>
<td></td>
<td>Response Templates</td>
<td>XSL</td>
<td>The response templates translate the XML response returned by a web service call into the format expected by the Connector Framework.</td>
</tr>
</tbody>
</table>
2.2 How an Event Connector Functions

In order to know how to build an event connector, you need to understand how one functions. The `connectorDeploy.xml` file, commonly referred to as the *connector descriptor file*, defines the connector in Enterprise Manager. When the connector is installed, the contents of the connector descriptor file are examined by Enterprise Manager to determine what fields are required by the connector and what templates to use when exchanging XML messages with the web service.

When the operator configures the connector, the configuration page is generated based on the information in the connector descriptor file. Typically this includes fields that contain the URL and credentials to use when connecting to the web service. After the operator specifies the required fields, they click the Ok button to complete the configuration process. When the button is clicked, the configuration values are saved and any setup or initialization service operations defined by the connector are performed.

---

**Note:** There are four optional service operations that can be specified. The setup and initialize operations are performed when the first instance of a connector is configured. These are used to perform any setup required to enable the connector. The other two operations are uninitialize and cleanup. These operations are used to undo anything that was done by the setup or initialize operations. Both of these operations are performed when the last instance of a connector is deleted.

---

If the configuration completed successfully, the connector is marked as completed and is ready to use. If an error occurs, the configuration changes will be saved but the connector will not be marked as completed. Someone must investigate why it failed and address the problem. Once the problem has been addressed, they can go back to the configuration page and click Ok to have perform setup_INITIALIZATION AGAIN.

Once the connector is marked as completed, you can set up incident rules to create an event in the external application whenever certain events occur. For example you could set up a rule to create an event if any of your database tablespaces exceed a specified threshold. Whenever the rule is set up, you specify what target(s) the rule applies to, what criteria to use to trigger the rule, and the event connector to invoke.

Whenever the criteria are met and the rule fires, the Connector Framework invokes the event connector to generate the XML required to create an event in the remote application. The Connector Framework creates the request XML by performing an XSLT translation on the Enterprise Manager event data using the `createEvent` request template defined by the connector. The Connector Framework sends the generated create request to the web service URL configured for the `createEvent` service. The web service creates an event in the external application and sends a response. When the Connector Framework gets the response, it uses the `createEvent` response template to translate the new event identifier into the format expected by the Connector Framework. When the Connector Framework gets the response, it persists the external event identifier with the event.

Whenever something occurs that causes the event in Enterprise Manager to be updated, the Connector Framework uses the `updateEvent` request template to translate the Enterprise Manager event data format into an XML update request that it sends to the web service. The web service updates the event in the external application and sends a response. When the Connector Framework gets the response, it uses the `updateEvent` response template to translate the response into a format expected by the
Connector Framework. This response is used by the Connector Framework to verify that the update was completed successfully.

When the event is eventually resolved and goes into a cleared status, the Connector Framework performs a normal `updateEvent` operation with the `SeverityCode` field set to CLEAR.

**Note:** The current release only supports outbound operations (sending events to external applications). The support for inbound (importing) external events into Enterprise Manager may be considered for a future release.

### 2.3 Prerequisites

You must have a good understanding of the XML, XSD, and XSLT technologies because you will be required to generate several XML and XSLT files during the course of building a connector. It is highly recommended that you familiarize yourself with these technologies before attempting to build a connector.

### 2.4 Extracting Schema Files

To create the ticketing and event connectors, you will need access to the schema files that define the format of the different files. The schema files are located in the Extensibility Development Kit (EDK). To install the EDK, go to the **Setup** menu, select **Extensibility**, then **Development Kit**. This page gives instructions for downloading and installing the EDK. Review the Requirements section and verify the prerequisites have been met before attempting to install the EDK. Once the prerequisites are confirmed, install the EDK as directed in the Deployment section.

The schema files are located in the `emMrsXsds.jar` file in the `emSDK` directory. To access the files, you will need to extract them using the `jar` command or any other utility that understands the jar file format. Use the following command to extract the files using the `jar` command from the EDK installation directory:

```
$JAVA_HOME/bin/jar xvf emSDK/emMrsXsds.jar
```

Table 2–2 shows the location of the extracted schema files. This table will be referenced in the different sections where the schema files are discussed.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectorDeploy.xsd</td>
<td>oracle/sysman/emSDK/core/connector/common</td>
</tr>
<tr>
<td>EMEvent.xsd</td>
<td>oracle/sysman/emSDK/core/connector/eventConnector</td>
</tr>
<tr>
<td>EMEventResponse.xsd</td>
<td>oracle/sysman/emSDK/core/connector/eventConnector</td>
</tr>
<tr>
<td>SelfUpdateManifest.xsd</td>
<td>oracle/sysman/emSDK/core/selfupdate/model</td>
</tr>
<tr>
<td>setupResponse.xsd</td>
<td>oracle/sysman/emSDK/core/connector/eventConnector</td>
</tr>
<tr>
<td>initialize_response.xsd</td>
<td>This file is not available in the EDK. See Example C–16, “initialize_response.xsd” in Appendix C.</td>
</tr>
<tr>
<td>uninitialize_response.xsd</td>
<td>This file is not available in the EDK. See Example C–17, “uninitialize_response.xsd” in Appendix C.</td>
</tr>
</tbody>
</table>
2.5 Building an Event Connector

Now that you understand how a connector functions, you are now ready to start the process of building your event connector. To build your connector, you will need to follow the instructions specified in the sections listed below:

- Determining Connector Functionality
- Developing Required Template Files
- Defining the Connector Descriptor File

2.5.1 Determining Connector Functionality

Before you can build a connector, you need to analyze your requirements and determine what functionality to include in the connector. This section assists you in determining what templates you will require for your event connector. When you are done with this section, you should have a list of the templates that need to be implemented for your connector.

There are some templates that are required and must be included in every event connector. You have no choice but to include these templates. There are other templates that are optional that may be included in the connector if deemed necessary. Table 2–3 lists the possible templates that can be defined for an event connector. The Description column in this table explains the functionality provided by the template. You will need to analyze the functionality provided by the optional templates and determine which ones you need to include in your connector. Once you complete this analysis, you should have a list of templates that you need to provide. Your list will be comprised of the required templates plus the optional templates that you have selected for inclusion.

Note: The optional templates are not used for most connectors. The only time you will want to use the optional template is when you need to make a web service call to set something up in the external application. An example of something that is done during initialization is the registration of the connector in the external application.

<table>
<thead>
<tr>
<th>Table 2–3 Possible Event Templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
</tr>
<tr>
<td>setup request</td>
</tr>
<tr>
<td>setup response</td>
</tr>
<tr>
<td>initialize request</td>
</tr>
<tr>
<td>initialize response</td>
</tr>
<tr>
<td>createEvent request</td>
</tr>
<tr>
<td>createEvent response</td>
</tr>
<tr>
<td>updateEvent request</td>
</tr>
</tbody>
</table>
You will also need to determine the file name that you want to use for each template. There are no requirements on the template file names, but Oracle recommends that you use the following naming convention:

```xml
<methodName>_request.xml
<methodName>_request.xsl
<methodName>_response.xsl
```

Table 2–4 lists the recommended filenames for the different templates based on the suggested naming convention:

### Table 2–4  Recommended Template Filenames

<table>
<thead>
<tr>
<th>Template</th>
<th>Recommended Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup request</td>
<td>setup_request.xml</td>
</tr>
<tr>
<td>setup response</td>
<td>setup_response.xsl</td>
</tr>
<tr>
<td>initialize request</td>
<td>initialize_request.xml</td>
</tr>
<tr>
<td>initialize response</td>
<td>initialize_response.xsl</td>
</tr>
<tr>
<td>createEvent request</td>
<td>createEvent_request.xsl</td>
</tr>
<tr>
<td>createEvent response</td>
<td>createEvent_response.xsl</td>
</tr>
<tr>
<td>updateEvent request</td>
<td>updateEvent_request.xsl</td>
</tr>
<tr>
<td>updateEvent response</td>
<td>updateEvent_response.xsl</td>
</tr>
<tr>
<td>uninitialize request</td>
<td>uninitialize_request.xml</td>
</tr>
<tr>
<td>uninitialize response</td>
<td>uninitialize_response.xsl</td>
</tr>
<tr>
<td>cleanup request</td>
<td>cleanup_request.xml</td>
</tr>
</tbody>
</table>

#### 2.5.2 Developing Required Template Files

Now that you have identified the templates that you need to provide, the next step is to create the template files. It is highly recommended that you use XML/XSLT tools to generate and test the template files. You can create the files using a standard text editor but it will make the process much more difficult and time consuming. The tools catch format errors and allow you to test the templates before you package and install the connector in Enterprise Manager. This greatly reduces the number of corrections that you have to make to the installed connector. Each correction that you have to make to the connector requires that you uninstall the old connector, repackage and reinstall the new version of the connector.
The following subsections cover the steps required to create the different template files. You can ignore the sections that cover templates that are not targeted for your connector.

- setup/initialize/uninitialize/cleanup Request Template
- setup/initialize/uninitialize Response Template
- createEvent/updateEvent Request Template
- createEvent/updateEvent Response Template

### 2.5.2.1 setup/initialize/uninitialize/cleanup Request Template

There is no event XML data to translate so these templates must be defined as XML files instead of XSLT files. Since these templates are defined as XML files, you just need to get a sample XML that is used to perform the operation that is required and use it as the XML file.

### 2.5.2.2 setup/initialize/uninitialize Response Template

The response template is an XSLT file that is used to transform the response from the web service into the format expected by Enterprise Manager. The format of the response XML from the web service should be defined by the WSDL or by a schema provided by the web service. The format expected by Enterprise Manager is specified in the Initialize_response.xsd, Setup_response.xsd, and Uninitialize_response.xsd schema files. See Table 2–2 in the Extracting Schema Files section for the location of these schema files.

Example 2–1 shows a sample response file from the web service. Example 2–2 shows a sample XSLT template that is designed to transform the data to the format expected by Enterprise Manager. The XSLT template looks for a root element with a name of registerResponse that has the following namespace:

http://oracle.com/services/adapter-framework

It creates a SetupResponse root element with the following namespace:

http://xmlns.oracle.com/sysman/connector

It creates a child ConnectorVariable element that contains a VariableName element that it sets to REGISTRATION_ID. It also creates a VariableValue element and sets it to the identifier specified in the registrationId element. Example 2–3 shows the XML that was generated by performing the translation using an XSLT tool.

**Example 2–1 Input Response XML from Web Service**

```xml
<adap:registerResponse xmlns:adap="http://oracle.com/services/adapter-framework">
  <adap:registrationId>2834782347</adap:registrationId>
</adap:registerResponse>
```

**Example 2–2 Sample XSLT Template File**

```xml
<?xml version='1.0'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
                xmlns:adap='http://oracle.com/services/adapter-framework'>
  <xsl:template match="/adap:registerResponse">
    <SetupResponse xmlns='http://xmlns.oracle.com/sysman/connector'>
      <ConnectorVariable>
        <VariableName>REGISTRATION_ID</VariableName>
        <VariableValue><xsl:value-of
```
Example 2–3  File Resulting from Transformation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SetupResponse xmlns="http://xmlns.oracle.com/sysman/connector"
xmlns:adap="http://oracle.com/services/adapter-framework">
<ConnectorVariable>
  <VariableName>REGISTRATION_ID</VariableName>
  <VariableValue>2834782347</VariableValue>
</ConnectorVariable>
</SetupResponse>
```

2.5.2.3 createEvent/updateEvent Request Template

These templates are the most difficult to build because they are larger and more complicated than the other templates. The input XML for the templates contain data about the Enterprise Manager event that was created/updated. The XML that results from the transformation is the XML that will be sent to the web service to create or update an event in the external application.

Before you can build your template, you need to understand the format of the data being created/updated in the external application and the format of the data coming from Enterprise Manager. Understanding the format involves identifying the fields that are available and the values that can be entered in those fields.

You need to familiarize yourself with the format of the data that is specified to create/update an event in the external application. A good place to start is the WSDL or a schema file that defines the format of the data. You will also need to see sample create/update requests to see how the data is formatted. If samples are not available, you should be able to manually retrieve data for an existing event using a XML client tool. This should give you a good idea of what the data looks like.

Once you understand the data in the external application, you then need to study and understand the data coming from Enterprise Manager. The fields that are available in the Enterprise Manager event data are identified in the `EMEvent.xsd` schema file. See Table 2–2 in the Extracting Schema Files section for the location of the `EMEvent.xsd` schema file.

The schema file identifies the fields and tells the type of data in those fields but doesn't give a good indication of what data is actually present. To get a good idea of what the data looks like you need a sample XML file that was generated by Enterprise Manager. Appendix G, “Sample Event Data,” shows sample event transactions that were generated by Enterprise Manager.

Once you are familiar with the data on both ends, you need to determine the mapping that will be performed by each template (createEvent and updateEvent). This involves determining what fields you will specify in the request and the format of the data for those fields. You only have two choices on the source of the data. You can hard code the data or get it from the Enterprise Manager event data. You can make the mapping as sophisticated or simple as you like. Ultimately, you will need to determine what fields and settings make sense for your environment.

Once you have identified mappings, you are now ready to build the XSLT files. To start the first template, copy the XML shown in Example 2–4 into your editor where
you are building the XSLT file. This contains the basic skeleton that you will need to build your template. You will need to add your mapping logic in the designated location. The recommended approach is to build the createEvent template first and to test it thoroughly. Once you have verified that it works, you can make a copy of the template and use that as a baseline for the updateEvent template.

Appendix E, "Create Event Template Example," walks through an example that shows how to build the createEvent template.

**Example 2–4 Template Skeleton**

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:emcf="http://xmlns.oracle.com/sysman/connector">

    <xsl:template match="emcf:EMEvent">
        <!-- Add your mapping here -->
    </xsl:template>

</xsl:stylesheet>
```

**2.5.2.4 createEvent/update Event Response Template**

These templates are comprised of an XSLT file that is used to transform the response from the web service into the format expected by Enterprise Manager. The format of the response XML from the web service should be defined by the WSDL or by the schema used by the web service. The format expected by Enterprise Manager is specified in the EMEventResponse.xsd schema file. See Table 2–2 in the Extracting Schema Files section for the location of the EMEventResponse.xsd schema file.

Example 2–5 shows a sample response file from the web service. Example 2–6 shows a sample XSLT template that is designed to transform the data to the format expected by Enterprise Manager. The XSLT template looks for a root element with a name of createResponse that has a namespace of http://oracle.com/services/adapter-framework and a child element of return. It creates an EMEventResponse root element with a namespace of http://xmlns.oracle.com/sysman/connector and creates two child elements that are set depending on the contents of the identifier element in the input document. If the identifier element exists and contains data, it sets the SuccessFlag element to true and the ExternalEventId element to the identifier. If the identifier is not specified or is empty, it sets the SuccessFlag to false and creates an ErrorMessage element that is set to "Request to create an event in the external application failed". Example 2–7 shows the XML that was generated by performing the translation using an XSLT tool.

**Example 2–5 Input Response XML from Web Service**

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<adap:createResponse xmlns:adap="http://oracle.com/services/adapter-framework">
    <return>
        <identifier>abcd-1234-5678</identifier>
        <status>0</status>
    </return>
</adap:createResponse>
```

**Example 2–6 Sample XSLT Template File**

```xml
<?xml version='1.0' ?>
<xsl:stylesheet version='2.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
    xmlns:oracleaf='http://oracle.com/services/adapter-framework'
    xmlns:a='http://xmlns.oracle.com/sysman/connector'>
```
Example 2-7 File Resulting from Transformation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<a:EMEventResponse xmlns:a="http://xmlns.oracle.com/sysman/connector"
xmlns:oracleaf="http://oracle.com/services/adapter-framework">
  <a:SuccessFlag>true</a:SuccessFlag>
  <a:ExternalEventId>abcd-1234-5678</a:ExternalEventId>
</a:EMEventResponse>
```

Whenever you create your XSLT template, it is easier to copy an existing template and customize it to fit your situation. To customize the template shown above:

1. Replace the oracleaf namespace definition with the namespace that will be specified in the XML coming from your web service.
2. Change the template match attribute to reference the root element in the XML coming from your web service.
3. Change the name of the element where it gets the event identifier information when it checks for an identifier and where it creates the ExternalEventId element.
4. Once you create your file, you should run an XSLT tool to test your template by transforming data from a sample XML and verify that it generates the correct XML format.

2.5.3 Defining the Connector Descriptor File

Now that you have the templates created, it is time to create a connector descriptor file that defines the connector in Enterprise Manager. The connector descriptor XML file describes the connector metadata and the configuration properties of the connector, such as web service end points and authentication schema.

The key points to remember when constructing a descriptor are:

- The connector descriptor file name must be `connectorDeploy.xml`
- The XML file should adhere to the `connectorDeploy.xsd` schema.

See Table 2–2 in the Extracting Schema Files section for the location of the connectorDeploy.xsd schema file.
Refer to the sample `connectorDeploy.xml` in Appendix C, "Event Connector Samples" for reference implementation.

Table 2–5 lists the sections that comprise a connector descriptor and provides a summary of what each section does:

<table>
<thead>
<tr>
<th>Metadata Section</th>
<th>Required</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector Information</td>
<td>Yes</td>
<td>This section provides information about the connector that will be displayed at the UI.</td>
</tr>
<tr>
<td>Authentication</td>
<td>No</td>
<td>Specifies the authentication method and the parameters that are required to connect to the web service.</td>
</tr>
<tr>
<td>Service</td>
<td>Yes</td>
<td>This section is used to configure the URLs used to connect to the web service for the different service operations.</td>
</tr>
<tr>
<td>Template Registration</td>
<td>Yes</td>
<td>This section is used to define all of the templates that are used for the connector.</td>
</tr>
</tbody>
</table>

The following sections provide detailed information about the contents of the connector descriptor:

- **Connector Deploy Field Information**
- **Connector Information Section**
- **Sample Connector Information Section**
- **Authentication Section**
- **Sample Authentication Section**
- **Service Section**
- **Sample Service Section**
- **Template Registration Section**
- **Sample Template Registration Section**

### 2.5.3.1 Connector Deploy Field Information

Each section of the connector deployment file contains fields that provide specific information about the connector. The following sections contain detailed information about what fields are available and what data goes in those fields.

### 2.5.3.2 Connector Information Section

The Connector Information section provides information about the connector such as name, version, description, etc., that will be displayed at the UI. Table 2–6 lists the fields in this section and provides an explanation of each field.

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The connector name that will be displayed at the UI.</td>
</tr>
<tr>
<td>Version</td>
<td>Yes</td>
<td>The connector version that will be displayed at the UI.</td>
</tr>
<tr>
<td>EMCompatibleVersion</td>
<td>Yes</td>
<td>The earliest version of Enterprise Manager that is supported</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>The connector description that will be displayed at the UI</td>
</tr>
</tbody>
</table>
2.5.3.3 Sample Connector Information Section

Example 2–8 shows the information that is included in the Connector Information section. All of the fields in this section are contained in the ManagementConnector node.

Example 2–8 Connector Information Section Sample

```
<Name>SCOM 2012 Connector</Name>
<Version>12.1.0.1.0</Version>
<EMCompatibleVersion>12.1.0.1.0</EMCompatibleVersion>
<Description>Microsoft System Center Operations Manager 2012 Integration with Enterprise Manager</Description>
<Category>EventConnector</Category>
<NewTargetType>
  <TargetTypeName>scom_managed_host</TargetTypeName>
  <TargetTypeDisplayName>SCOM Managed Host</TargetTypeDisplayName>
  <DefaultTargetName>generic_scom_managed_host</DefaultTargetName>
  <DefaultTargetDisplayName>Generic SCOM Managed Host</DefaultTargetDisplayName>
</NewTargetType>
```

2.5.3.4 Authentication Section

The authentication section specifies the authentication method and the credentials that are required to connect to the web service. There are three possible authentication types that can be configured. If no authentication section is specified, no authentication will be performed when the connector connects to the web service. 

Table 2–7 lists the three possible authentication sections and the fields contained in each.

**Table 2–7 Authentication Fields**

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAPHeaderAuthentication</td>
<td>No</td>
<td>Specifies the credentials used to connect to the web service using SOAP header authentication.</td>
</tr>
<tr>
<td>*Username</td>
<td>Yes</td>
<td>The username to specify in the SOAP header</td>
</tr>
<tr>
<td>*Password</td>
<td>Yes</td>
<td>The password to specify in the SOAP header</td>
</tr>
<tr>
<td>*AuthVariable</td>
<td>No</td>
<td>Up to 20 other variables to pass in the SOAP header</td>
</tr>
<tr>
<td>*SOAPHeader</td>
<td>Yes</td>
<td>A string that serves as template for the SOAP header. It will be updated by substituting the user inputs for variables defined above in the designated location and bound with a HTTP request.</td>
</tr>
</tbody>
</table>
2.5.3.5 Sample Authentication Section

Example 2–9 shows the information that is included in the Authentication section. In this example, the authentication method is basic authentication. The operator would be required to provide SCOM Web Service Username and SCOM Web Service Password values when configuring the connector. The values entered by the operator would be passed in the basic authentication header for any requests that are sent to the web service.

Example 2–9 Authentication Section Sample

```xml
<HTTPBasicAuthentication>
  <Username required="true">
    <VariableName>Username</VariableName>
    <DisplayName>SCOM Web Service Username</DisplayName>
  </Username>
  <Password required="true">
    <VariableName>Password</VariableName>
    <DisplayName>SCOM Web Service Password</DisplayName>
  </Password>
</HTTPBasicAuthentication>
```

2.5.3.6 Service Section

This section is used to configure the URLs used to connect to the web service for the different service operations. Each entry that is defined in this section must also define the corresponding templates in the Template Registration section. There must be a separate Service section entry for each of the following operations:

- createEvent
- updateEvent

The following service operations are optional:

- setup
Building an Event Connector

- initialize
- uninitialize
- cleanup

**Note:** The service names in the connector descriptor should *exactly* match the names defined above and are case sensitive.

Table 2–8 lists the fields in the section and provides an explanation of each field:

**Table 2–8  Service Fields**

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Yes</td>
<td>This section allows you to specify configurations specific to the External System’s web services.</td>
</tr>
<tr>
<td>Method</td>
<td>Yes</td>
<td>Method defines one of the EM-specific service operation names.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For event connectors, it must be set to one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>initialize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createEvent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateEvent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>uninitialize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cleanup</td>
</tr>
<tr>
<td>WebServiceEndpoint</td>
<td>Yes</td>
<td>This field specifies the default web service endpoint string to be displayed in the web service section of the Management Connector page.</td>
</tr>
<tr>
<td>SOAPAction</td>
<td>No</td>
<td>The SOAPAction to specify when calling the web service</td>
</tr>
<tr>
<td>SOAPBindingType</td>
<td>No</td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP11HTTP_BINDING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP12HTTP_BINDING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP11HTTP_MTOM_BINDING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOAP12HTTP_MTOM_BINDING</td>
</tr>
</tbody>
</table>

**2.5.3.7 Sample Service Section**

**Example 2–10** shows both of the required Service sections and three optional sections. The URLs in the WebServiceEndpoint element are placed in a CDATA section to avoid conflicts with reserved XML characters. The values surrounded by square brackets [] need to be replaced by the operator on the configuration page.

For example, the default URL for the `createEvent` operation is:

http://[Hostname]:8080/services/SCOM/EventService

This value will be displayed in the Web Service End Points section of the configuration page next to the `createEvent` operation. The operator will need to replace [Hostname] with the hostname or IP address of the system where the web service is hosted.

**Example 2–10  Service Section Sample**

```xml
<Service>
  <Method>setup</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[Hostname]:8080/services/SCOM/EventService]]>
  </WebServiceEndpoint>
</Service>
```
2.5.3.8 Template Registration Section

This section is used to define all of the templates that were built in the previous chapter. **Table 2–9** lists the fields in the section and provides an explanation of each field.

**Table 2–9 Template Registration Fields**

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TemplateRegistration</td>
<td>Yes</td>
<td>This section defines up to 50 connector templates. Each template that you created in Developing Required Template Files needs to be defined here.</td>
</tr>
</tbody>
</table>
### Table 2–9 (Cont.) Template Registration Fields

<table>
<thead>
<tr>
<th>Section/Field</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>Yes</td>
<td>Name of the file that defines the template.</td>
</tr>
<tr>
<td>InternalName</td>
<td>Yes</td>
<td>Internal template name. It must be set to one of the following service method names. It is case sensitive so it must match exactly. setup initialize createEvent updateEvent uninitialize cleanup</td>
</tr>
<tr>
<td>TemplateName</td>
<td>Yes</td>
<td>Name to use in the UI when referencing this template.</td>
</tr>
<tr>
<td>TemplateType</td>
<td>Yes</td>
<td>There are two types of templates. One is Outbound and the other is Inbound. Outbound is used to generate XML that is being sent to the external web service and Inbound is used to transform incoming XML to the format expected by Enterprise Manager. Possible values for the template type are: InboundXSL OutboundXSL OutboundXML</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>Description of the template that will be displayed at the UI.</td>
</tr>
</tbody>
</table>

### 2.5.3.9 Sample Template Registration Section

Example 2–11 shows the different TemplateRegistration sections:

**Example 2–11 Template Registration Section Sample**

```
<TemplateRegistration>
    <FileName>setup_request.xml</FileName>
    <InternalName>setup</InternalName>
    <TemplateName>Setup Request</TemplateName>
    <TemplateType>OutboundXML</TemplateType>
    <Description>This is the request xml file for the setup method</Description>
</TemplateRegistration>

<TemplateRegistration>
    <FileName>setup_response.xsl</FileName>
    <InternalName>setup</InternalName>
    <TemplateName>Setup Response</TemplateName>
    <TemplateType>InboundXSL</TemplateType>
    <Description>This is the response xsl file for the setup method</Description>
</TemplateRegistration>

<TemplateRegistration>
    <FileName>setup_request.xml</FileName>
    <InternalName>initialize</InternalName>
    <TemplateName>Initialize Request</TemplateName>
    <TemplateType>OutboundXML</TemplateType>
    <Description>This is the request xml file for the initialize method</Description>
</TemplateRegistration>

<TemplateRegistration>
    <FileName>createEvent_request_2012.xsl</FileName>
    <InternalName>createEvent</InternalName>
    <TemplateName>Create Event Request</TemplateName>
    <TemplateType>OutboundXSL</TemplateType>
    <Description>This is the request xsi file for the createEvent method</Description>
```
2.5.3.10 Complete Connector Deployment File

Example C–1 in Appendix C shows the complete connector deployment file that includes the samples shown in the preceding sections. Figure 2–1 shows an example of the connector configuration page that is displayed for the sample deployment file. The image has been labeled to show where the fields were defined in the deployment file.
2.6 Packaging and Deploying the Event Connector

To deploy the connector, Enterprise Manager uses the Self Update feature. This feature, which can be accessed through the console, provides the ability to import the connector into the Enterprise Manager environment. To deploy the connector complete the following:

1. Prepare the connector jar file

   Package all XML and XSLT template files as a .jar file

   `<name>_connector.jar
   --&gt; connectorDeploy.xml
   --&gt;template1.xml
   --&gt;template2.xsl
   ...` 
   `--&gt;templateN.xsl`

2. Prepare the manifest file

   Table 2–10 lists the Key attributes of self update manifest files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityType</td>
<td>Value is core_connector</td>
</tr>
<tr>
<td>EntityTypeVersion</td>
<td>Current release version. Value=12.1.0.1.0</td>
</tr>
<tr>
<td>Version</td>
<td>Version number of the connector. Must be set to the value specified in the ManagementConnector/Version node in the connectorDeploy.xml file.</td>
</tr>
<tr>
<td>Attribute @Name=connector_type</td>
<td>Connector type name. Must be set to the value specified in the ManagementConnector/Name node in the connectorDeploy.xml file</td>
</tr>
</tbody>
</table>
The `SelfUpdateManifest.xsd` schema file defines the format of the manifest file.

See Table 2–2 in the Extracting Schema Files section for the location of the `SelfUpdateManifest.xsd` schema file.

The following example shows the code for the `connector_manifest.xml` file:

**Example 2–12 Manifest File Sample**

```xml
<EntityInstanceId
xmlns="http://www.oracle.com/EnterpriseGridControl/SelfUpdateManifest">
<EntityInstance
xmlns="http://www.oracle.com/EnterpriseGridControl/SelfUpdateManifest"
EntityTypeId="12.1.0.1.0" EntityType="core_connector"
Maturity="PRODUCTION"
Vendor="Oracle" PluginId="oracle.sysman.core">
<Description>
<![CDATA[ Microsoft SCOM 2012 Connector - 12.1.0.1.0 ]]>]
</Description>
<AttributeList>
<Version>12.1.0.1.0</Version>
<Attribute Name="connector_type" Value="SCOM 2012 Connector"
Label="SCOM 2012 Connector"/>
<Attribute Name="connector_category" Value="EventConnector"
Label="Event Connector"/>
</AttributeList>
<Readme><![CDATA[
The Oracle Management Connector for Microsoft System Center Operations Manager (SCOM) 2012 enables you to forward Enterprise Manager alerts to SCOM 2012. The integration is a uni-directional connection so information only flows from Enterprise Manager to SCOM. State changes in Enterprise Manager are reflected in SCOM. However, if you change the state of the alert in SCOM, the change is not reflected in Enterprise Manager.

The connector requires the installation of an Oracle SCOM agent on a Windows system with connectivity to the RMS server system. In addition to the agent, an Oracle SCOM Web Service must also be installed. The web service must be installed on a system that has connectivity to the system where the agent is installed and the Enterprise Manager server system. The web service is Java based and can be installed on any Windows or UNIX platform that supports Oracle JRE version 6.

This connector only supports SCOM 2012. There is a separate connector that must be used with versions of SCOM 2007.

Some configuration changes are required in SCOM to allow alerts to be created by Enterprise Manager. A management pack must be imported and an account must be set up that can be used to access the SCOM API.

Change Logs:

12.1.0.1.0
- Initial Release
]]>
</Readme>
</EntityInstance>
</EntityInstanceId>
```

### Table 2–10 (Cont.) Self Update Manifest File Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute @Name=connector_category</td>
<td>Category type can be TicketingConnector or EventConnector</td>
</tr>
<tr>
<td>ArchiveList</td>
<td>This element contains the list of archives that are part of connector setup. Generally there will be single connector jar but for some special implementation there may be additional jars (adapter or agent). In these cases, the connector specific jar should be first one in the defined list. This is mandatory requirement.</td>
</tr>
</tbody>
</table>
3. Configure the emedk tool

   The emedk tool can be configured by following instructions from the Enterprise Manager user interface. From the Setup menu, select Extensibility, then Development Kit.

4. Prepare the self-update archive

   This requires the connector jar file and the manifest file for the connector. To prepare self-update, call the following utility to create a self update archive file:

   ```
   edkutil prepare_update
   -manifest "manifest xml"
   -archivedir "archives directory"
   -out "output file or directory"
   [-typexml "update type xml"]
   ```

   Table 2–11 describes the options available with the utility:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-manifest</td>
<td>Self update manifest file that describes the update.</td>
</tr>
<tr>
<td>-archivedir</td>
<td>Directory containing the archive files specified in the manifest file.</td>
</tr>
<tr>
<td>-out</td>
<td>Directory or filename for the self update archive. If a directory is specified, the filename is autogenerated.</td>
</tr>
<tr>
<td>-typexml</td>
<td>Optional path to the update type xml</td>
</tr>
</tbody>
</table>

   The following example creates a self update archive in the /u01/sar directory based on the manifest file /u01/connector/connector_manifest.xml. The archives referred to in connector_manifest.xml are picked from the directory /u01/connector/archives.

   ```
   edkutil prepare_update
   -manifest /u01/connector/connector_manifest.xml
   -archivedir /u01/connector/archives
   -out /u01/sar/sample_connector.zip
   ```

5. Import the connector archive to Enterprise Manager by calling any one of the following emcli commands:

   ```
   emcli import_update
   -file=\ file\ 
   -omslocal
   ```

   or
These commands import a Self Update archive file into Enterprise Manager. On successful import, the update is displayed on the Self Update Home in downloaded status for further action. Table 2–12 describes the connector archive command options.

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-file</td>
<td>The complete pathname of the update archive file</td>
</tr>
<tr>
<td>-omslocal</td>
<td>The flag specifying that the file is accessible from the OMS</td>
</tr>
<tr>
<td>-host</td>
<td>The target name for a host target where the file is available</td>
</tr>
<tr>
<td>-credential_set_name</td>
<td>The set name of the preferred credential stored in the repository for the host target. Can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- HostCredsNormal Default unprivileged credential set</td>
</tr>
<tr>
<td></td>
<td>- HostCredsPriv Privileged credential set</td>
</tr>
<tr>
<td>-credential_name</td>
<td>The name of a named credential stored in the repository. This option must be specified along with -credential_owner option.</td>
</tr>
<tr>
<td>-credential_owner</td>
<td>The owner of a named credential stored in the repository. This option must be specified along with -credential_name option.</td>
</tr>
</tbody>
</table>

The following paragraphs provide some examples of the use of the emcli command:

**Example 1**

Imports the file update1.zip. The file must be present on the OMS host. In a multiple OMS setup, the request can be processed by any OMS, so the file should be accessible from the OMS processing the request. This usually means that the file must be kept on a shared location that is accessible from all OMS.

```
emcli import_update
-file=/u01/common/update1.zip
-omslocal
```

**Example 2**

Imports the file update1.zip that is present on the host1.example.com host. The host must be a managed host target in Enterprise Manager and the agent on this host must be up and running. The preferred unprivileged credentials for host host1.example.com are used to retrieve the remote file.

```
emcli import_update
-file=/u01/common/update1.zip
-host=\ host1.example.com\  
-credential_set_name=\ HostCredsNormal\ 
```

**Example 3**

Imports the file update1.zip that is present on the host1.example.com host. The host must be a managed host target in Enterprise Manager and the agent on this
host must be up and running. The named credentials \ host1_creds\ owned by user \ admin1\ are used to retrieve the remote file.

```emcli
emcli import_update
-file=\ /u01/common/update1.zip\n-host=\ host1.example.com\n-credential_name=\ host1_creds\n-credential_owner=\ admin1\n```

6. Apply the connector using one of the following methods:

- From the Cloud Control console:
  a. Go to Self-Update Home page. The connector will be shown as downloaded.
  b. Select the connector row and click **Apply** to deploy the connector.

- From the command line, run the following `emcli list` command to determine the identifier of the connector that was just imported:

  ```emcli
  emcli list -resource=Updates -bind="et_name = 'core_connector'"
  ```

  The output of the command would look like this example:

<table>
<thead>
<tr>
<th>Status</th>
<th>Category</th>
<th>Type</th>
<th>Version</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied</td>
<td>Ticketing</td>
<td>Remedy Service</td>
<td>12.1.0.1.0</td>
<td>123456789ABCDE</td>
</tr>
<tr>
<td></td>
<td>Desk Connector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applied</td>
<td>Event</td>
<td>HP OMU Connector</td>
<td>12.1.0.3.0</td>
<td>11223344AABBCC</td>
</tr>
<tr>
<td></td>
<td>Connector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applied</td>
<td>Ticketing</td>
<td>Remedy Service</td>
<td>12.1.0.3.0</td>
<td>1A2B3C4D5E6F7G</td>
</tr>
<tr>
<td></td>
<td>Connector</td>
<td>Desk 7.6 Connector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applied</td>
<td>Ticketing</td>
<td>CASD Connector</td>
<td>12.1.0.3.0</td>
<td>55443322CCBBAA</td>
</tr>
<tr>
<td></td>
<td>Connector</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  Note the ID for the connector that was just imported. You will need to select the ID for the connector that was just imported and supply to the `emcli apply_updates` command listed below:

  ```emcli
  emcli apply_updates -id=<ID>
  ```
Building a Data Exchange Connector

A Data Exchange Connector is a JMS server-based integration vehicle that helps you to build a bi-directional data exchange setup between Enterprise Manager and other management systems. The Data Exchange Connector architecture is based on open standards such as Java Message Service (JMS) and XML. This helps in facilitating easy extensibility and interoperability.

The data exchange environment necessitates creation of a data exchange hub and data exchange sessions. This chapter explains the key concepts, components, and features involved in the data exchange process.

Also provided are specific steps to integrate Enterprise Manager with Oracle Business Activity Monitoring Server (OBAM).

This chapter discusses these topics:
- Introduction
- Data Exchange Concepts
- Setting up a Data Exchange Connector
- Integrating Enterprise Manager with Oracle BAM
- Using an OC4J as a Data Exchange Hub
- Tips and Troubleshooting Information
- Suggested Reading

3.1 Introduction

Typically, an enterprise may have Enterprise Manager monitoring most of the systems and services within it. However, other monitoring systems or external management systems such as the Oracle BAM server might also exist within the business environment of the enterprise. These management systems and Enterprise Manager might be collecting monitoring information that is different, yet related to the same business application. It is imperative for the business that Enterprise Manager and these external management systems co-exist and interact seamlessly.

A Data Exchange Connector effectively addresses this communication requirement by transferring data in XML format using JMS Topic or Queue messages. This is made possible by creating a data exchange hub and a data exchange session.

Figure 3–1 provides an architectural overview of the Data Exchange Connector.
3.1.1 Enterprise Manager and External Management System

Table 3–1 explains the data requirements and purpose of data exchange between Enterprise Manager and an external management system.

<table>
<thead>
<tr>
<th>Data Exchange</th>
<th>Requirement</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Enterprise Manager to external management system</td>
<td>Metric Data</td>
<td>To use the data to correlate with business indicators and events that business applications send to an external management system. Metric data also includes Service-level Agreement (SLA) and target status data besides raw metrics.</td>
</tr>
<tr>
<td></td>
<td>Alert Data</td>
<td>Better reporting, event notification, and corrective actions.</td>
</tr>
<tr>
<td>From external management system to Enterprise Manager</td>
<td>Business Indicators</td>
<td>Better reporting and topology analysis.</td>
</tr>
<tr>
<td></td>
<td>Business Events</td>
<td>Generate a comprehensive SLA in the Enterprise Manager environment.</td>
</tr>
</tbody>
</table>

Integrating the two systems using Data Exchange Connector helps the systems to complement each other and serve business requirements effectively and economically.

3.1.2 Data Forwarding Frequency Options and Modes

The following list explains the normal process followed when sending data from Enterprise Manager to external management systems.

- Real-time metric, availability, and SLA values are forwarded, as well as historical data.
- Historical data is forwarded for the last 24 hours, 7 days, and 31 days. Historical data for the target status is sent as the percentage of time in which the target is available during the time period.
Metric data is forwarded in batches at scheduled intervals. In each batch, a maximum of 100 data points can be sent. An interval of two seconds is maintained between subsequent forwarding to reduce the JMS server load.

For a given metric, all new data points in the interval are sent to the external system. If there are no new values, no data is sent. For the initial forwarding, data points since the previous one hour are considered. One hour is the default interval, but you can configure this to a different time interval.

For example, if an outbound session is scheduled from 9:00 a.m. to 9:00 p.m. with an interval frequency of 30 minutes, initially (at 9:00 a.m.) metric values collected between 8:00 a.m. and 9:00 a.m. are forwarded. Subsequently, the metric values received in that interval are sent. So at 9:30 a.m., metric values received between 9:00 a.m. and 9:30 a.m., and at 10:00 a.m. metric values received between 9:30 a.m. and 10:00 a.m., are forwarded.

Alerts are sent without latency. Each outbound message only has one alert embedded in it.

Forwarded Service-level Agreement (SLA) data is the SLA value for the selected time period. For a 24-hour scenario, if an outbound session with an SLA metric is scheduled at January 15th at 4:00 p.m., the value forwarded is the SLA value from January 14th at 4:01 pm to January 15th at 4:00 p.m.

### 3.2 Data Exchange Concepts

The following sections explain the major concepts that you must understand to successfully set up a data exchange environment between Enterprise Manager and an external management system.

#### 3.2.1 Data Exchange Hub

A data exchange hub is a JMS-compliant server that acts as the conduit between Enterprise Manager and an external management system. Data is sent and received between external systems and Cloud Control through such a hub. The hub should be configured with known JMS destination information (Outbound JMS Destinations) so that the messages can be sent and retrieved seamlessly. The Data Exchange Hub page shows a list of existing Data Exchange hubs and their related JNDI Service Provider URLs, provided that at least one hub has already been created. Examples of a hub are WebLogic Server (WLS), Oracle Containers for JEE (OC4J), and so forth.

See Also: Creating a Data Exchange Hub

#### 3.2.2 Inbound Data Exchange Session

An inbound data exchange session is created to receive business indicators, events, or both from the data source of an external system to Enterprise Manager.

See Also: "Creating an Inbound Data Exchange Session" on page 3-28

#### 3.2.3 Outbound Data Exchange Session

An outbound data exchange session is created to send metric values, alerts, target availability, or a combination of them from Enterprise Manager to an external system.
The data can be sent in either of the following formats:

- Normalized message format
- Denormalized message format

### 3.2.3.1 Normalized Message Format

In this format, data is sent in two phases.

- **Session Setup Phase** — Meta information for targets and metrics such as target name, target type, metric name, and metric column are sent along with their GUIDs when the session is created in Enterprise Manager Cloud Control.

- **Session Execution Phase** — Actual metrics are sent when the session is executed. They are tagged with the GUIDs to avoid sending redundant meta information for every message, thereby keeping the wire footprint low.

This message format is effective if the external system is backed by a persistence store, such as a database, so that it can retrieve the metadata by joining the tables when rendering the charts or reports based on GUIDs.

### 3.2.3.2 Denormalized Message Format

In this format, target and metric meta information is sent along with every message in the session execution phase. No messages are sent during the session setup phase. This message format is effective if the external system is not backed by a persistence store. Though each message repeats the meta information, digesting the data for charting and reporting is easier.

---

**See Also:** "Creating an Inbound Data Exchange Session" on page 3-28

---

### 3.2.4 Message Schemas

To correctly parse and interpret the contents, it is imperative for the external system to understand the syntax and semantics of the XML messages embedded in the JMS destinations. The schema of the message varies depending on the message format (normalized or denormalized).

The same JMS destinations are used for both formats; therefore, sessions with different message formats should not run concurrently because it confuses the consumer of these messages. Oracle recommends that the sessions with different formats be run exclusively.

### 3.2.5 Data Source

Data source is a logical representation of an external system source from which business indicators or events are retrieved. A data source definition represents the following:

- The structure and schema of the business content (business indicators) received from the external system.
- The transport (JMS destinations) information by way of which the external data (business events and indicators) is received.
- Associated target in Enterprise Manager to which the external data (business events and indicators) is associated.
3.2.6 Average Data

Besides selecting raw metrics, you can also select average metrics for intervals of 24 hours, 7 days, and 31 days. The following conditions apply:

- The default is raw metrics per session.
- You cannot mix and match raw data or different levels of average data per session. For a given session, all data must be raw, one of the average types, or of the same average level.
- Alerts are always sent real-time and have no bearing for the average selection.
- You can also send SLA and target availability using average data, expressed as a percentage.

3.3 Setting up a Data Exchange Connector

1. From Enterprise Manager Cloud Control, click Setup.
2. Click Extensibility.
3. Click Data Exchange.

   The Data Exchange page appears as shown below.

   ![Data Exchange Setup](image)

4. Set up the Data Exchange Connector.

   The following sections provide information required to set up a Data Exchange Connector:
   - Creating a Data Exchange Hub
   - Creating an Outbound Data Exchange Session
   - Creating an Inbound Data Exchange Session

3.3.1 Creating a Data Exchange Hub

The following JMS servers are certified and supported:

- WebLogic Server 8.1 series and above
- OC4J 10.1.3.1 series
- Oracle Enterprise Service Bus (OESB) 10.1.3.1 series
- OC4J 10.1.2.0 series
Do the following to create a data exchange hub:

1. In the Data Exchange: Data Exchange Hub page click Create. The Create Data Exchange Hub page appears.
   - Specify a unique name for the Data Exchange hub.
   - Provide the JNDI Service Provider URL for the Data Exchange hub.
   - Select a context factory name from the list according to the matrix in Table 3–2.
   - Provide the user credentials to access this hub.
2. Configure the JMS server with the required JMS topic names and/or queue names.
3. Click OK to save the configuration and return to the Data Exchange: Data Exchange Hub page.

After you create a hub for data exchange, you can set up an outbound or inbound data exchange session.

### Table 3–2 Context Factory Name

<table>
<thead>
<tr>
<th>Hub Corresponds to ...</th>
<th>Context Factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebLogic Server</td>
<td>weblogic.jndi.WLInitialContextFactory</td>
</tr>
<tr>
<td>10.1.2.X OC4J</td>
<td>com.evermind.server.rmi.RMIInitialContextFactory</td>
</tr>
<tr>
<td>10.1.3.X OC4J</td>
<td>oracle.j2ee.rmi.RMIInitialContextFactory</td>
</tr>
<tr>
<td>Non-WebLogic Server and non-OC4J JMS Server</td>
<td>For third-party servers, select Other and provide a factory name in the Enter JNDI Initial Context Factory Name field.</td>
</tr>
</tbody>
</table>

- d. Provide the user credentials to access this hub.

### 3.3.2 Using a Third-Party JMS Server as a Data Exchange Hub

**Caution:** The use of third-party JMS servers is uncertified and unsupported.

Although such usage is uncertified and unsupported, it is possible to use third-party JMS servers as a Data Exchange hub in a development or test scenario by performing the following procedure.

1. Copy your JMS server’s JMS client libraries to the following location:
   
   `$ORACLE_HOME/middleware/oms/sysman/archives/emgc/deployments/EMGC_DOMAIN/emgc.ear/APP-INF/lib`

2. Restart Cloud Control after copying the .jar file(s).
3. Create the JMS destinations according to the procedures specified for your JMS server. Refer to the list of Topics and Queues on page 3-9.

### 3.3.3 Creating an Outbound Data Exchange Session

To create an outbound data exchange session, specify the input provided in the following procedure for the respective pages of the setup wizard.

1. From the Data Exchange page, click the **Outbound Data Exchange Session** link.
2. Click **Create**. The Session Setup step of the wizard appears.
   
   a. Ensure that you have access to at least one Data Exchange hub that is configured with the topics to receive data from Enterprise Manager. To set up a Data Exchange hub, see Creating a Data Exchange Hub.
   
   b. Specify a unique name for this outbound data exchange session in the Name field.
   
   c. Select a Data Exchange hub from the list of hubs already created and listed alphabetically. By default, the first hub in the list is selected.
   
   d. In the Users field, enter the name of an Oracle BAM user or a comma-separated list of Oracle BAM users (for example: user1, user2) on the external system who can access the data being sent. The name or name of these users for this session will be associated with the Oracle BAM server. The receiving system must interpret and enforce security. If you leave this field blank, no user information is sent.
   
   e. For the Destination Type drop-down, select one of the messaging models to send or receive messages from Cloud Control. Queues provide point-to-point messaging interaction based on Queue logic. Topics provide broadcast model-based subscription methodology.
      
      If the session uses Topic as the Destination Type, all messages for this session are sent on the corresponding pre-defined topics. For Queues, all messages are sent on the pre-defined Queues. Mix-and-match of destination types for a particular session is not supported.
   
   f. Select either a **Normalized** or **Denormalized** message format. For more information on these formats, see [Outbound Data Exchange Session](#).
   
   g. In the Metrics Granularity drop-down, select whether you want to send raw metrics data through an outbound session, or whether you want to send average data for 24 hours, 7 days, or 31 days.
   
   h. Click **Next**. The Select Targets and Metrics/Alerts/Availability step of the wizard appears.
   
3. Click **Add**. The Add Target page of this wizard step appears.
4. Click the search icon to invoke the Search and Select Targets pop-up, choose a target that is to be part of the Data Exchange transactions being set up, then click **Select**. A list of all metrics and objects for the target now appear on the Add Target page.
5. Specify which metrics and alerts you want to send during the outbound session for the target. Specify which metrics and alerts you want to send during the outbound session for the target. You can use the batch selection buttons if you want to select or deselect all targets and all alerts.
   
   - You need to select **Send Metric Values** for the Availability metric for target availability information.
You cannot select the Send Alerts for the Service-Level Agreement (SLA) metric. This metric is only applicable and available for service targets. For example, host targets do not have this metric.

Some metrics may require you to specify an object. For example, you can qualify the Filesystem Space Available (%) metric by the name of a particular mount point.

6. Click OK to save your selections and return to the Select Targets and Metrics/Alerts/Availability step.

You can now view or edit your selections. You can also use your selections as a template for another target by clicking the Add-like icon.

7. Click Next if you are satisfied with your metric values and alerts selections. The Schedule step of the wizard appears. Select one of the following scheduling choices:

   ■ Schedule Later — You can defer scheduling and subsequently schedule the session from the Outbound Data Exchange Session sub-page after you click Finish in the Review step of the wizard.

   ■ Schedule Now — Choose one of the following sub-types:
     - One Time (Immediately): If you select this option, the session runs once just when you finish creating it.
     - One Time (Later): If you select this option, you need to specify a time zone and a start date and time for the session.
     - Repeating: For this default option, you need to specify the time zone and the start time. Additionally, you can specify the frequency type and interval at which you want the session to run, and whether it should be repeated indefinitely or until a specified time and date.

8. Click Next or Review to go to the Review step of the wizard.

   If you need to make changes, click Back until you reach the step you need to change. Otherwise, go to the next step.

9. Click Finish. The Outbound Data Exchange Session sub-page reappears and shows your newly created session and its status in the table.

   Before the job finishes executing, you can either view the schedule by clicking the View Schedule link in the Actions column and then stop the execution if desired, or you can stop the execution immediately by clicking Stop.

3.3.4 Outbound JMS Destinations

Predefined topic and queue names are used to send data from Enterprise Manager to external systems through the hub. You should configure the data exchange hub with the JMS destination information specified in Table 3-3 through Table 3-9. It is not mandatory to define both topics and queues. If you always want to use the topics, for example, you can remove the queue definitions or not initially create them, and vice versa.

Example - Configuring JMS Destinations for WebLogic Server

You can use any JMS-compliant server with the Data Exchange Connector as described in this example.

To configure the JMS destinations for WebLogic Server, do the following:
1. Use the pre-packaged WLST python scripts available in the $ORACLE_HOME/sysman/bam directory.

2. Set the proper CLASSPATH before going to the next step. You can set the CLASSPATH by running setWLSEnv.sh found under ORACLE_HOME (typically in the middleware/wlserver_10.3/server/bin directory).

3. Use configEMSYSJMSSystemResource.py found in the directory in step 1 to create the required JMS topics and queues:

   java weblogic.WLST configEMSYSJMSSystemResource.py <jndi provider URL> <username> <password> <WLS server name>

   Example:

   java weblogic.WLST configEMSYSJMSSystemResource.py "t3://localhost:7001" weblogic welcome1 AdminServer

A successful run of the script produces the following components:

- **Resources and Destinations:**
  - EMSYSJMSServer for JMS Server
  - EMSYSJMSSystemResource for JMS System Resource
  - EMSYSJMSServerDeployment for sub-deployment

- **Connection Factories:**
  - jms/EMSYSTopicConnectionFactory
  - jms/EMSYSQueueConnectionFactory

- **Topics:**
  - jms/EMSYSTargetsTopic
  - jms/EMSYSMetricsTopic
  - jms/EMSYSAlertsDataTopic
  - jms/EMSYSMetricsDataTopic
  - jms/EMSYSSecurityFilterTopic
  - jms/EMSYSTargetStatusTopic
  - jms/EMSYSTargetSLATopic
  - jms/EMSYSMetricsDataLast24HoursTopic
  - jms/EMSYSMetricsDataLast7DaysTopic
  - jms/EMSYSMetricsDataLast31DaysTopic
  - jms/EMSYSTargetStatusLast24HoursTopic
  - jms/EMSYSTargetStatusLast7DaysTopic
  - jms/EMSYSTargetStatusLast31DaysTopic
  - jms/EMSYSTargetSLALast24HoursTopic
  - jms/EMSYSTargetSLALast7DaysTopic
  - jms/EMSYSTargetSLALast31DaysTopic

- **Queues:**
  - jms/EMSYSTargetsQueue
- \texttt{jms/EMSYSMetricsQueue}
- \texttt{jms/EMSYSAlertsDataQueue}
- \texttt{jms/EMSYSMetricsDataQueue}
- \texttt{jms/EMSYSSecurityFilterQueue}
- \texttt{jms/EMSYSTargetStatusQueue}
- \texttt{jms/EMSYSTargetSLAQueue}
- \texttt{jms/EMSYSMetricsDataLast24HoursQueue}
- \texttt{jms/EMSYSMetricsDataLast7DaysQueue}
- \texttt{jms/EMSYSMetricsDataLast31DaysQueue}
- \texttt{jms/EMSYSTargetStatusLast24HoursQueue}
- \texttt{jms/EMSYSTargetStatusLast7DaysQueue}
- \texttt{jms/EMSYSTargetStatusLast31DaysQueue}
- \texttt{jms/EMSYSTargetSLALast24HoursQueue}
- \texttt{jms/EMSYSTargetSLALast7DaysQueue}
- \texttt{jms/EMSYSTargetSLALast31DaysQueue}

4. Use \texttt{EMSYSJMSSystemResource.py} to remove and clean up the JMS destinations the script \texttt{configEMSYSJMSSystemResource.py} created. CLASSPATH should also be set as defined in Step 2 for the following command:

\begin{verbatim}
java weblogic.WLST deleteEMSYSJMSSystemResource.py <jndi provider URL> <username> <password>
\end{verbatim}

Example:

\begin{verbatim}
java weblogic.WLST deleteEMSYSJMSSystemResource.py 't3://localhost:7001'
weblogic welcome1
\end{verbatim}

The JMS destinations shown in Table 3–3 are used for an outbound Data Exchange session.

\textbf{Table 3–3 JMS Destination for Targets}

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>\texttt{jms/EMSYSTopicConnectionFactory} or \texttt{jms/EMSYSQueueConnectionFactory}</td>
</tr>
<tr>
<td>Destination Name</td>
<td>\texttt{jms/EMSYSTargetsTopic} or \texttt{jms/EMSYSTargetsQueue}</td>
</tr>
<tr>
<td>JMS Message Type</td>
<td>Text message</td>
</tr>
<tr>
<td>Description</td>
<td>Target metadata information, such as target name and type are sent on this destination.</td>
</tr>
</tbody>
</table>
Note:

- For outbound sessions using Topic as Destination Type, jms/EMSYSTopicConnectionFactory and all topic versions (ms/EMSYSTargetsTopic and so forth) are used.
- For Destination Type as Queue, jms/EMSYSQueueConnectionFactory and queue versions (jms/EMSYSTargetsQueue and so forth) are used.

Table 3–4  JMS Destination for Metrics

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory</td>
</tr>
<tr>
<td>Destination Name</td>
<td>jms/EMSYSMetricsTopic or jms/EMSYSMetricsQueue</td>
</tr>
<tr>
<td>JMS Message Type</td>
<td>Text message</td>
</tr>
<tr>
<td>Description</td>
<td>Metric metadata information, such as metric name, column, and target type are sent on this destination.</td>
</tr>
</tbody>
</table>

Table 3–5  JMS Destination for Raw or Average Metric Data

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory</td>
</tr>
<tr>
<td>Destination Name</td>
<td>- Raw: jms/EMSYSMetricsDataTopic or jms/EMSYSMetricsDataQueue</td>
</tr>
<tr>
<td></td>
<td>- Last 24 Hours: jms/EMSYSMetricsDataLast24HoursTopic or jms/EMSYSMetricsDataLast24HoursQueue</td>
</tr>
<tr>
<td></td>
<td>- Last 7 Days: jms/EMSYSMetricsDataLast7DaysTopic or jms/EMSYSMetricsDataLast7DaysQueue</td>
</tr>
<tr>
<td></td>
<td>- Last 31 Days: jms/EMSYSMetricsDataLast31DaysTopic or jms/EMSYSMetricsDataLast31DaysQueue</td>
</tr>
<tr>
<td>JMS Message Type</td>
<td>Text message</td>
</tr>
<tr>
<td>Description</td>
<td>This destination is used to send raw or average metric values.</td>
</tr>
</tbody>
</table>

Table 3–6  JMS Destination for Raw or Average Target Status

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory</td>
</tr>
</tbody>
</table>
Table 3–6 (Cont.) JMS Destination for Raw or Average Target Status

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Name</td>
<td>• Raw: jms/EMSYSTargetStatusTopic or jms/EMSYSTargetStatusQueue</td>
</tr>
<tr>
<td></td>
<td>• Last 24 Hours: jms/EMSYSTargetStatusLast24HoursTopic or jms/EMSYSTargetStatusLast24HoursQueue</td>
</tr>
<tr>
<td></td>
<td>• Last 7 Days: jms/EMSYSTargetStatusLast7DaysTopic or jms/EMSYSTargetStatusLast7DaysQueue</td>
</tr>
<tr>
<td></td>
<td>• Last 31 Days: jms/EMSYSTargetStatusLast31DaysTopic or jms/EMSYSTargetStatusLast31DaysQueue</td>
</tr>
</tbody>
</table>

| JMS Message Type    | Text message                                                                |
| Description         | This destination is used to send raw (numeric value) or average (percentage) target status information. |

Table 3–7 JMS Destination for Security Filter

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory</td>
</tr>
<tr>
<td>Destination Name</td>
<td>jms/EMSYSSecurityFilterTopic</td>
</tr>
<tr>
<td>JMS Message Type</td>
<td>Text message</td>
</tr>
<tr>
<td>Description</td>
<td>Security filter information is sent on this destination.</td>
</tr>
</tbody>
</table>

Table 3–8 JMS Destination for Alert Data

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory</td>
</tr>
<tr>
<td>Destination Name</td>
<td>jms/EMSYSAlertsDataTopic</td>
</tr>
<tr>
<td>JMS Message Type</td>
<td>Text message</td>
</tr>
<tr>
<td>Description</td>
<td>Alerts are sent on this destination.</td>
</tr>
</tbody>
</table>

Table 3–9 JMS Destination for Raw or Average SLA Data

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionFactory Name</td>
<td>jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory</td>
</tr>
</tbody>
</table>
3.3.5 Outbound Message Schema

The following sections explain the outbound message schema. The schema varies depending on whether the message format is normalized or denormalized.

To avoid regressions and conflicts between different type of metric data, the XML element and target names differ for raw versus historical data. Table 3–10 shows the differences for these based on the type of granularity generated.

### Table 3–10 Raw Versus Historical Data

<table>
<thead>
<tr>
<th>Granularity</th>
<th>Metric Element Name</th>
<th>Target Status Element Name</th>
<th>SLA Target Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>MetricData</td>
<td>TargetStatus</td>
<td>TargetSLA</td>
</tr>
<tr>
<td>Last 24 Hours</td>
<td>MetricDataLast24Hours</td>
<td>TargetStatusLast24Hours</td>
<td>TargetSLALast24Hours</td>
</tr>
<tr>
<td>Last 7 Days</td>
<td>MetricDataLast7Days</td>
<td>TargetStatusLast7Days</td>
<td>TargetSLALast7Days</td>
</tr>
<tr>
<td>Last 31 Days</td>
<td>MetricDataLast31Days</td>
<td>TargetStatusLast31Days</td>
<td>TargetSLALast31Days</td>
</tr>
</tbody>
</table>

3.3.5.1 Normalized Message Format

The schema for outgoing messages for a normalized format is as follows:

#### Normalized Target Message

For each selected target, corresponding target metadata information is sent to the external system during the session setup phase. The schema of these messages is as follows:

### Table 3–11 Normalized Target Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/Target</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundTarget.xsd</td>
</tr>
</tbody>
</table>
Normalized Metric Message
For each selected metric, corresponding metric metadata information is sent to the external system during the session setup. The schema of these messages is as follows:

Table 3–12 Normalized Metric Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/Metric</td>
</tr>
</tbody>
</table>

Normalized Target Message
For each selected metric, corresponding metric metadata information is sent to the external system during the session setup. The schema of these messages is as follows:

Table 3–11 (Cont.) Normalized Target Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>jms/EMSYSTargetsTopic or jms/EMSYSTargetsQueue</td>
</tr>
</tbody>
</table>
| Schema              | <?xml version="1.0" encoding="UTF-8"?>
<xs:schema

targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
xmlns:id="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified" attributeFormDefault="unqualified">

<xs:element name="EMSYSData" type="de:EMSYSDataType"/>

<xs:complexType name="EMSYSDataType">
<xs:sequence>
<xs:element name="Target" type="de:TargetType"
minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="TargetType">
<xs:all>
<xs:element name="TargetName" type="xs:string"/>
<xs:element name="TargetType" type="xs:string"/>
<xs:element name="TargetGUID" type="xs:string"/>
</xs:all>
</xs:complexType>
</xs:schema>

Sample
<de:EMSYSData
xmlns:id="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
<Target>
<TargetName>/ade/foo_core9/oracle.stacd16.us.oracle.com_home</TargetName>
<TargetType>oc4j</TargetType>
<TargetGUID>852464ac3e4176460458297faaffe926</TargetGUID>
</Target>
<Target>
<TargetName>stacd16.us.oracle.com</TargetName>
<TargetType>host</TargetType>
<TargetGUID>00645a665bfd9b72b2a6bb6ef49606b0</TargetGUID>
</Target>
</de:EMSYSData>
Setting up a Data Exchange Connector

Building a Data Exchange Connector

Normalized Security Filter Message

External systems that consume data from Enterprise Manager can enforce access control based on the session name. This can be achieved by capturing the security filter. The schema of these security filter messages is as follows:

Table 3–12 (Cont.) Normalized Metric Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundSecurityFilter.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYMetricsTopic or jms/EMSYMetricsQueue</td>
</tr>
<tr>
<td>Schema</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
 xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- Define the root element -->
  <xs:complexType name="EMSYSDataType">
    <!-- Define the Metric Type -->
    <xs:complexType name="MetricType">
      <xs:all>
        <xs:element name="MetricName" type="xs:string"/>
        <xs:element name="MetricType" type="xs:string"/>
        <xs:element name="MetricGUID" type="xs:string"/>
        <xs:element name="MetricColumn" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      </xs:all>
    </xs:complexType>
  </xs:complexType>
</xs:schema>
```

Sample

```xml
<de:EMSYSData
 xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
  <Metric>
    <MetricColumn>DiskActivityavgserv</MetricColumn>
    <MetricGUID>6ca028d5078fe542b2ee0c1b013727d7</MetricGUID>
    <TargetType>host</TargetType>
    <MetricName>DiskActivity</MetricName>
  </Metric>
  <Metric>
    <MetricColumn>cpuUtil</MetricColumn>
    <MetricGUID>0c71a1afac2d7199013837da35522c08</MetricGUID>
    <TargetType>host</TargetType>
    <MetricName>Load</MetricName>
  </Metric>
</de:EMSYSData>
```

Normalized Security Filter Message

External systems that consume data from Enterprise Manager can enforce access control based on the session name. This can be achieved by capturing the security filter. The schema of these security filter messages is as follows:
### Table 3–13 Normalized Security Filter Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/SecurityFilter</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundMetric.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSSecurityFilterTopic or jms/EMSYSSecurityFilterQueue</td>
</tr>
</tbody>
</table>

#### Schema

```xml
<xs:schema
    targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
    xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <!-- Define the root element -->
  <xs:complexType name="EMSYSDataType">
    <xs:sequence>
      <xs:element name="SecurityFilter" type="de:SecurityFilterType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Define the Security Filter Type -->
  <xs:complexType name="SecurityFilterType">
    <xs:all>
      <xs:element name="SessionName" type="xs:string"/>
      <xs:element name="UserName" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:schema>
```

#### Sample

```xml
<de:EMSYSData
    xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
  <SecurityFilter>
    <SessionName>LoanSession</SessionName>
    <UserName>LoanAdminUser</UserName>
  </SecurityFilter>
</de:EMSYSData>
```

### Table 3–14 Normalized Metric Data Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/MetricData</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundNormalizedMetricsData.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSMetricsDataTopic or jms/EMSYSMetricsDataQueue</td>
</tr>
</tbody>
</table>

#### Normalized Metric Data Message

In the normalized message format, the metrics are sent along with the GUIDs to avoid sending meta information for every message. The schema of this metric message is as follows:
Normalized Alert Message

In the normalized message format, the alerts are sent along with the GUIDs to avoid sending meta information for every message. The schema of this alert message is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <xs:complexType name="EMSYSDataType">
    <xs:sequence>
      <xs:element name="MetricData" type="de:MetricDataType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MetricDataType">
    <xs:all>
      <xs:element name="SessionName" type="xs:string"/>
      <xs:element name="TargetGUID" type="xs:string"/>
      <xs:element name="MetricGUID" type="xs:string"/>
      <xs:element name="Timestamp" type="xs:dateTime"/>
      <xs:element name="Value" type="xs:float" minOccurs="0"/>
      <xs:element name="StringValue" type="xs:string" minOccurs="0"/>
      <xs:element name="KeyValue" type="xs:string" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:schema>
```

Sample

```xml
<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
  <MetricDataLast24Hrs>
    <MetricGUID>df8067b7d515747434b58d69230b8451</MetricGUID>
    <Value>0.67</Value>
    <Timestamp>07/13/2012 14:44:42</Timestamp>
    <SessionName>Session1</SessionName>
    <TargetGUID>00645a665bfdf9b72b2a6bb6ef49606b0</TargetGUID>
  </MetricDataLast24Hrs>
</de:EMSYSData>
```

Table 3–14 (Cont.) Normalized Metric Data Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
</table>
| Schema             | <?xml version='1.0' encoding='UTF-8'?>...
| Sample             | <de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">

Normalized Alert Message

In the normalized message format, the alerts are sent along with the GUIDs to avoid sending meta information for every message. The schema of this alert message is as follows:
### Table 3–15 Normalized Alert Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/Alert</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundNormalizedAlertsData.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSAlertsDataTopic or jms/EMSYSAlertsDataQueue</td>
</tr>
</tbody>
</table>

**Sample**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
    <!-- Define the root element -->
    <xs:complexType name="EMSYSDataType">
        <xs:sequence>
            <xs:element name="Alert" type="de:AlertType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <!-- Define the Alert Type -->
    <xs:complexType name="AlertType">
        <xs:all>
            <xs:element name="SessionName" type="xs:string" minOccurs="0"/>
            <xs:element name="TargetGUID" type="xs:string" minOccurs="0"/>
            <xs:element name="MetricGUID" type="xs:string" minOccurs="0"/>
            <xs:element name="Value" type="xs:float" minOccurs="0"/>
            <xs:element name="Message" type="xs:string" minOccurs="0"/>
            <xs:element name="KeyValue" type="xs:string" minOccurs="0"/>
        </xs:all>
    </xs:complexType>
</xs:schema>
```

Sample

```xml
<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
    <Alert>
        <MetricGUID>0c71a1afac2d719013837da35522c08</MetricGUID>
        <Value>25.35</Value>
        <Message>CPU Utilization is 25.35%, crossed warning (15) or critical (95) threshold.</Message>
        <Severity>Warning</Severity>
        <Timestamp>07/13/2012 14:59:42</Timestamp>
        <SessionName>Session9</SessionName>
        <TargetGUID>0645a665b9b2a6bb6ef49606b0</TargetGUID>
    </Alert>
</de:EMSYSData>
```
List of Severities

- CLEAR
- INFO
- WARNING
- CRITICAL
- AGENT UNREACHABLE CLEAR
- AGENT UNREACHABLE START
- BLACKOUT END
- BLACKOUT START
- METRIC ERROR END
- METRIC ERROR START

Normalized Target Availability Message

The schema of a normalized target availability information message is as follows:

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/TargetStatus</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundNormalizedTargetStatus.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSTargetStatusTopic or jms/EMSYSTargetStatusQueue</td>
</tr>
<tr>
<td>Schema</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema

xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <!-- Define the root element -->
  <xs:complexType name="EMSYSDataType">
    <xs:sequence>
      <xs:element name="TargetStatus" type="de:TargetsStatusType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Define the Target Status Type -->
  <xs:complexType name="TargetStatusType">
    <xs:all>
      <xs:element name="SessionName" type="xs:string"/>
      <xs:element name="TargetGUID" type="xs:string"/>
      <xs:element name="Status" type="xs:integer"/>
      <xs:element name="Timestamp" type="xs:dateTime"/>
    </xs:all>
  </xs:complexType>
</xs:schema>
Possible status values are provided in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Target is up and reachable</td>
</tr>
<tr>
<td>-1</td>
<td>Target is down</td>
</tr>
<tr>
<td></td>
<td>Metric error</td>
</tr>
<tr>
<td></td>
<td>Agent is down</td>
</tr>
<tr>
<td>0</td>
<td>Blackout</td>
</tr>
<tr>
<td></td>
<td>Target is not monitored</td>
</tr>
<tr>
<td></td>
<td>Target is unknown</td>
</tr>
</tbody>
</table>

**Normalized Target SLA Message**

The schema of a normalized target SLA information message is as follows:

**Table 3-17 Normalized Target SLA Message**

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/TargetSLA</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundNormalizedTargetSLA.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSystTargetSLATopic or jms/EMSystTargetSLAQueue</td>
</tr>
</tbody>
</table>
Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for Normalized Outbound Target Status message -->
<xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"/>
<!-- Define the root element -->
<xs:element name="EMSYSData" type="de:EMSYSDataType"/>
<!-- Define the root type -->
<xs:complexType name="EMSYSDataType">
<!-- Zero or more TargetSLA elements -->
<xs:sequence>
  <xs:element name="TargetSLA"
    type="de:TargetSLAType" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<!-- Define the TargetSLA Type -->
<xs:complexType name="MetricDataType">
<xs:all>
  <xs:element name="SessionName" type="xs:string"/>
  <xs:element name="TargetGUID" type="xs:string"/>
  <xs:element name="SLA" type="xs:integer"/>
  <xs:element name="Timestamp" type="xs:dateTime"/>
</xs:all>
</xs:complexType>
</xs:schema>
```

Table 3–17  Normalized Target SLA Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td></td>
</tr>
</tbody>
</table>

<!-- Schema for Normalized Outbound Target Status message -->
<xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"/>
<!-- Define the root element -->
<xs:element name="EMSYSData" type="de:EMSYSDataType"/>
<!-- Define the root type -->
<xs:complexType name="EMSYSDataType">
<!-- Zero or more TargetSLA elements -->
<xs:sequence>
  <xs:element name="TargetSLA"
    type="de:TargetSLAType" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<!-- Define the TargetSLA Type -->
<xs:complexType name="MetricDataType">
<xs:all>
  <xs:element name="SessionName" type="xs:string"/>
  <xs:element name="TargetGUID" type="xs:string"/>
  <xs:element name="SLA" type="xs:integer"/>
  <xs:element name="Timestamp" type="xs:dateTime"/>
</xs:all>
</xs:complexType>
</xs:schema>
3.3.5.2 Denormalized Message Format

Following sections describe the schema for the outgoing messages for denormalized format.

**Denormalized Metric Data Message**

Schema of a denormalized metric data message is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for Normalized Outbound Target Status message -->
<xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
  xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
  xmlns:xsi="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- Define the root element -->
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <!-- Define the root type -->
  <xs:complexType name="EMSYSDataType">
    <!-- Zero or more TargetSLA elements -->
    <xs:sequence>
      <xs:element name="TargetSLA" type="de:TargetSLAType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Define the TargetSLA Type -->
  <xs:complexType name="MetricDataType">
    <xs:all>
      <xs:element name="SessionName" type="xs:string"/>
      <xs:element name="TargetGUID" type="xs:string"/>
      <xs:element name="SLA" type="xs:integer"/>
      <xs:element name="Timestamp" type="xs:dateTime"/>
    </xs:all>
  </xs:complexType>
</xs:schema>
<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
  <TargetStatusLast7Days>
    <Status>1</Status>
    <Timestamp>07/11/2012 16:21:53</Timestamp>
    <SessionName>Session1</SessionName>
    <TargetGUID>00645a665bf97b2a6bb6ef49606b0</TargetGUID>
  </TargetStatusLast7Days>
</de:EMSYSData>
```
### Table 3–18 Denormalized Metric Data Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/MetricData</td>
</tr>
<tr>
<td>Schema File Location</td>
<td><code>$ORACLE_HOME/sysman/bam/OutboundDenormalizedMetricsData.xsd</code></td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSMetricsDataTopic or jms/EMSYSMetricsDataQueue</td>
</tr>
<tr>
<td>Schema</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <!-- Define the root element -->
  <xs:complexType name="EMSYSDataType">
    <xs:sequence>
      <xs:element name="MetricData" type="de:MetricDataType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Define the Metric Data Type -->
  <xs:complexType name="MetricDataType">
    <xs:all>
      <xs:element name="SessionName" type="xs:string" minOccurs="0"/>
      <xs:element name="TargetName" type="xs:string" minOccurs="0"/>
      <xs:element name="TargetType" type="xs:string" minOccurs="0"/>
      <xs:element name="MetricName" type="xs:string" minOccurs="0"/>
      <xs:element name="MetricColumn" type="xs:string" minOccurs="0"/>
      <xs:element name="Timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="Value" type="xs:float" minOccurs="0"/>
      <xs:element name="StringValue" type="xs:string" minOccurs="0"/>
      <xs:element name="KeyValue" type="xs:string" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:schema>
```

**Sample**

```xml
<de:EMSYSData
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"/>
<MetricDataLast24Hours>
  <SessionName>Session1</SessionName>
  <TargetName>OC4J 10.1.3</TargetName>
  <TargetType>generic_service</TargetType>
  <MetricName>Usage Value</MetricName>
  <Timestamp>2001-12-17T09:30:47-05:00</Timestamp>
  <Value>3.14159</Value>
</MetricDataLast24Hours>
</EMSYSData>
```
Denormalized Alert Message

Schema of a denormalized alert message is as follows:

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/Alert</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundDenormalizedAlertsData.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSAlertsDataTopic or jms/EMSYSAlertsDataQueue</td>
</tr>
</tbody>
</table>
| Schema              | `<xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
  xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <!-- Define the root element -->
  <xs:complexType name="EMSYSDataType">
    <xs:sequence>
      <xs:element name="Alert" type="de:AlertType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Define the Alert Type -->
  <xs:complexType name="AlertType">
    <xs:all>
      <xs:element name="SessionName" type="xs:string"/>
      <xs:element name="TargetName" type="xs:string"/>
      <xs:element name="TargetType" type="xs:string"/>
      <xs:element name="MetricName" type="xs:string"/>
      <xs:element name="MetricColumn" type="xs:string" minOccurs="0"/>
      <xs:element name="Timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="Severity" type="xs:string" minOccurs="0"/>
      <xs:element name="Value" type="xs:float" minOccurs="0"/>
      <xs:element name="Message" type="xs:string" minOccurs="0"/>
      <xs:element name="KeyValue" type="xs:string" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:schema>` |
Denormalized Target Availability Message
Schema of a denormalized target availability message is as follows:

Table 3–20  Denormalized Target Availability Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<de:EMSYSData
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/>
  <Alert>
    <SessionName>Session9</SessionName>
    <TargetName>OC4J 10.1.3</TargetName>
    <TargetType>generic_service</TargetType>
    <MetricName>Usage Value</MetricName>
    <Value>25.35</Value>
    <Message>CPU Utilization is 25.35%, crossed warning (15) or critical (95) threshold.</Message>
    <Severity>Warning</Severity>
    <Timestamp>07/13/2012 14:59:42</Timestamp>
  </Alert>
</de:EMSYSData>
```
Denormalized Target SLA Message

The schema of a denormalized SLA information message is as follows:

### Table 3–20 (Cont.) Denormalized Target Availability Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
</table>
| Schema              | `<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
  <!-- Define the root element -->
  <xs:complexType name="EMSYSDataType">
    <xs:sequence>
      <xs:element name="TargetStatus" type="de:TargetStatusType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>` |
| Sample              | `<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
  <TargetStatus>
    <Status>1</Status>
    <Timestamp>07/11/2012 16:21:53</Timestamp>
    <SessionName>Session1</SessionName>
    <TargetName>OC4J 10.1.3</TargetName>
  </TargetStatus>
</de:EMSYSData>` |

### Table 3–21 Denormalized Target SLA Message

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Expression</td>
<td>EMSYSData/TargetSLA</td>
</tr>
<tr>
<td>Schema File Location</td>
<td>$ORACLE_HOME/sysman/bam/OutboundDenormalizedTargetSLA.xsd</td>
</tr>
<tr>
<td>Destination</td>
<td>jms/EMSYSTargetSLATopic or jms/EMSYSTargetSLAQueue</td>
</tr>
</tbody>
</table>
3.3.6 Tuning Outbound Session Message Parameters

You can tune outbound session message parameters using the emctl command, as shown in Table 3–22.

<table>
<thead>
<tr>
<th>Elements and Sample</th>
<th>Description</th>
</tr>
</thead>
</table>
| Schema              | <?xml version="1.0" encoding="UTF-8"?>
                        <!-- Schema for Denormalized Outbound Target Status message -->
                        <xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
                        xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/
                        xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        elementFormDefault="qualified" attributeFormDefault="unqualified">
                        <!-- Define the root element -->
                        <xs:element name="EMSYSData" type="de:EMSYSDataType"/>
                        <!-- Define the root type -->
                        <xs:complexType name="EMSYSDataType">
                        <!-- zero or more target status elements -->
                        <xs:sequence>
                            <xs:element name="TargetSLA"
                                type="de:MetricDataType"
                                minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                        </xs:complexType>
                        <!-- Define the Target Status Type -->
                        <xs:complexType name="TargetSLAType">
                        <xs:all>
                            <xs:element name="SessionName"
                                type="xs:string"/>
                            <xs:element name="TargetName"
                                type="xs:string"/>
                            <xs:element name="TargetType"
                                type="xs:string"/>
                            <xs:element name="SLA" type="xs:integer"/>
                            <xs:element name="Timestamp"
                                type="xs:dateTime"/>
                        </xs:all>
                        </xs:complexType>
                        </xs:schema>
                        
                        Sample
                        <de:EMSYSData
                        xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/">
                        <TargetSLA>
                            <SLA>100</SLA>
                            <Timestamp>07/11/2012 16:21:53</Timestamp>
                            <SessionName>Session1</SessionName>
                            <TargetName>OC4J 10.1.3</TargetName>
                            <TargetType>generic_service</TargetType>
                        </TargetSLA>
                        </de:EMSYSData>

Note: For the parameters to become effective, you need to restart OMS after setting the properties.
### Setting up a Data Exchange Connector

**Example 3–1 Command Syntax for Tuning Outbound Session Message Parameters**

```plaintext
emctl {set property|get property}
{oracle.sysman.core.dataExchange.MaxDataPointsPerMessage
 oracle.sysman.core.dataExchange.IntervalBetweenMessage
 oracle.sysman.core.dataExchange.FirstDatasetWindow}
```

You need to restart OMS after the properties have been set to be effective.

### 3.3.7 Creating an Inbound Data Exchange Session

To create an inbound data exchange session, specify the input provided in the following procedure for the respective pages of the setup wizard.

1. From the Data Exchange page, click the **Inbound Data Exchange Session** link.

2. Click **Create**. The Session Setup step of the wizard appears.

   a. Ensure that you have access to at least one Data Exchange hub that is configured with the topics to receive data from Enterprise Manager. To set up a Data Exchange hub, see **Creating a Data Exchange Hub**.

   b. Specify a unique name for this inbound data exchange session in the Name field.

   c. Select a Data Exchange hub from the list of hubs already created and listed alphabetically. By default, the first hub in the list is selected.

   Incoming business events are associated with a corresponding business KPI if you import the KPI to Cloud Control. If not, the event is associated with the special built-in metric, named `ExternalAlertMetric`.

   d. Click **Next**. The Select Business Events/Indicators step of the wizard appears.

3. If you need to send business events to Cloud Control, click **Add Business Events**. Otherwise, skip to step 4. The Select Business Events/Indicators: Add Business Events page appears.

   a. Specify the name of an existing data source from which the event will be retrieved. The data source names within a session should be unique.

   b. Optionally check **Apply XSL Transformation for incoming messages** to apply an XSL style sheet to the incoming message, the conversion of which is shown in **Figure 3–2**. A multi-line text box appears when you click **Show**, where you can insert an XSL document. If you choose this option, click **Check Syntax** to check the accuracy of your entry before proceeding.

---

**Table 3–22 Tuneable Outbound Session Message Parameters**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Default Value</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle.sysman.core.dataExchange.MaxDataPointsPerMessage</td>
<td>100</td>
<td>Number of metric data points within a message.</td>
</tr>
<tr>
<td>oracle.sysman.core.dataExchange.IntervalBetweenMessage</td>
<td>2 seconds</td>
<td>Time gap between subsequent JMS messages in seconds.</td>
</tr>
<tr>
<td>oracle.sysman.core.dataExchange.FirstDatasetWindow</td>
<td>60 minutes</td>
<td>When sending the first message, date for the past first set data window is sent. The unit is in minutes.</td>
</tr>
</tbody>
</table>
The most common usage for XSLT conversion involves incoming messages transporting business KPIs or events. When the KPIs or events do not produce the expected message (schema), you can apply the XSL at the Cloud Control end rather than changing the message format itself.

**Figure 3–2  XSLT Conversion of XML Source Document**

- **c.** Specify JMS destination details for incoming events. You need to specify the ConnectionFactory, the destination from which data is retrieved, and an optional Durable Subscriber Name (only needed for topics, not for queues) so that all messages pertaining to the topic go to the specified subscriber. Specifying a Durable Subscriber for topics prevents you from losing any incoming events.

  To ensure that an authenticated connection will be created between Cloud Control and the Data Exchange hub, you can specify a user name and password so that the connection can be established with these credentials. Click **Test Connection** to verify that your input is valid.

- **d.** Associate the business events with a target that Cloud Control is monitoring. You can associate business events with any Enterprise Manager monitored target.

  The target drop-down lists all the available target types, with the Generic Service target type being the default. If you want to choose the business event associated with a specific target that is not available in the list, add the target from the Targets page and then restart the procedure of creating an inbound session.

- **e.** Click **OK** to save your configuration, then view your input in a tabular format and edit your selections if necessary.

  You can also use your selections as a template for another target by clicking **Add-like**.

4. If you need to send business indicators to Cloud Control, click **Add Business Indicators**. Otherwise, skip to step 5. The Select Business Events/Indicators: Add Business Indicators page appears.
a. Specify the name of an existing external data source from which the indicators will be retrieved. The data source names within a session should be unique. If the XML message sent from the data source is namespace-enabled, select the check-box indicating this, and also specify the fully-qualified namespace.

b. Optionally check **Apply XSL Transformation for incoming messages** to apply an XSL style sheet to the incoming message. A multi-line text box appears after you click **Show**, where you can insert an XSL document. If you choose this option, click **Check Syntax** to check the accuracy of your entry before proceeding.

c. Specify the business indicators that need to be sent to Cloud Control by clicking **Add Indicator**. The corresponding metric name for the business indicator is `<Source Name>_<Indicator Name>`. All indicators can only have numeric values.

d. Specify JMS destination details for incoming indicators. You need to specify the ConnectionFactory, the destination from which data is retrieved, and an optional Durable Subscriber Name (only needed for topics, not for queues) so that all messages pertaining to the topic go to the specified subscriber. Specifying a Durable Subscriber topic prevents you from losing any incoming indicators.

To ensure that an authenticated connection will be created between Cloud Control and the Data Exchange hub, you can specify a user name and password so that the connection can be established with these credentials. Click **Test Connection** to verify that your input is valid.

e. Associate the business indicators with a target that Cloud Control is monitoring. Unlike business events, which can be associated with any target type instance, business indicators can be associated only with instances that are of the Service target type.

The target drop-downs list all the available target types, with the Generic Service target type being the default. If you want to choose the business event associated with a specific target that is not available in the list, add the target from the Targets page and then restart the procedure of creating an inbound session.

f. Click **OK** to save your input, then view your input in a tabular format and edit your selections if necessary.

You can also use your selections as a template for another target by clicking **Add-like**.

5. Click **Next** if you are satisfied with the configuration. The Schedule step of the wizard appears. Select one of the following scheduling choices:

- **Schedule Later** — You can defer scheduling and subsequently schedule the session from the Outbound Data Exchange Session sub-page after you click Finish in the Review step of the wizard.

- **Schedule Now** — Choose one of the following sub-types:
  - **One Time (Immediately)**: If you select this option, the session runs once just when you finish creating it.
  - **One Time (Later)**: If you select this option, you need to specify a time zone and a start date and time for the session.
  - Repeating: For this default option, you need to specify the time zone and the start time. Additionally, you can specify the frequency type and
Setting up a Data Exchange Connector

3.3.8 Inbound JMS Destinations

Unlike the outbound data exchange setup wherein pre-defined topics and queues are used to send Enterprise Manager data, no pre-defined topics or queues are used to receive business performance indicators and events.

However, you should configure the JMS topics or queues used for the data sources in the JMS server used for inbound data exchange session.

3.3.9 Inbound Message Schemas

The following sections define the inbound message schemas. Samples messages are provided along with each schema.

3.3.9.1 Inbound Indicators Schema

After creating the session, the sender can forward the data in XML format using the data exchange hub through the JMS destinations defined in the inbound data exchange session.

Messages can be either namespace qualified or unqualified. If the messages are namespace qualified, the namespace should be entered during the data source setup time.

Qualified XML Message Sample

```xml
<po:PurchaseOrder xmlns:po="http://acme.com/Orders">
  <OrderAmount>5000</OrderAmount>
  <NoOfItems>15</NoOfItems>
</po:PurchaseOrder>
```

Unqualified XML Message Sample

```xml
<PurchaseOrder>
  <OrderAmount>5000</OrderAmount>
  <NoOfItems>15</NoOfItems>
</PurchaseOrder>
```

3.3.9.2 Message Semantics

The incoming messages should follow the semantics provided below:

- The local name of the top-level element should be same as the data source name as in Example 3–2.
- If the message is qualified, the namespace should be defined during the data source setup time.
One or more indicators can be sent as child elements within this element as in Example 3–2.

A sub-element with the Timestamp as the name has special semantics. If a sub-element with the Timestamp name exists, the indicators are inserted with this Timestamp value. If no Timestamp element exists, the current time is used when inserting the indicator into the repository.

For example, if the request is received as follows with the Timestamp sub-element, the indicators are inserted with this timestamp (2012-09-30 17:43:19.474):

```xml
  <OrderAmount>5000</OrderAmount>
  <NoOfItems>15</NoOfItems>
  <Timestamp>2012-10-30 17:43:19.474</Timestamp>
</po: PurchaseOrder>
```

If no Timestamp sub-element is present, the indicators are inserted to the repository with the current timestamp at which they are received.

Example 3–2 Data Source Scenario
You create a Data Source for the incoming business indicators with the Data Source name Order. You add the following three KPIs:

- OrderAmount
- NoOfItems
- Credit

In this case, the incoming XML message should be in the following format:

```xml
<Order>
  <OrderAmount>35</OrderAmount>
  <NoOfItems>102</NoOfItems>
  <Credit>72</Credit>
  <Timestamp>2007-01-16 16:29:00.978</Timestamp>
</Order>
```

**Note:** In the example, the local name of the top-level element should be same as the Data source name <Order>.

Also, the indicators such as Credit are sent as child elements with the same name.

Message Element Defaults
- If TargetName and TargetType are part of the message, they should match the target name and type for the associated target (for that data source).
- If TargetName is not part of the message, it defaults to the target to which the data source was associated.
- If TargetType is not part of the message, it defaults to the target type of the target.
- If Timestamp is not included in the message, it defaults to the current timestamp.
- If Category is not included in the message, it defaults to the category GenericExternalAlertMetric.
- If MetricName is not included in the message, it defaults to the Alert metric.
- **ProducerID** is optional for the categories `GenericExternalAlertMetric` and `Metric`.

  However, producer ID is needed for user-defined metrics. In this case, **ProducerID** should be same as the metric author.

### 3.3.9.3 Inbound Alert Schema

External systems can send their own alerts/events to Enterprise Manager for display in the Enterprise Manager pages and be computed as part of SLA.

This schema is available in the following location:

\$/ORACLE_HOME/sysman/bam/InboundEvents.xsd

The schema of the incoming Alert message is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/InboundEvents/
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/InboundEvents/
xmlns:xsi="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <!-- Define the Alert element -->
  <xs:element name="Alert" type="de:AlertType"/>
  <!-- Define the Alert Type -->
  <xs:complexType name="AlertType">
    <xs:all>
      <xs:element name="TargetType" type="xs:string" minOccurs="0"/>
      <xs:element name="TargetName" type="xs:string" minOccurs="0"/>
      <xs:element name="Category" type="xs:string" minOccurs="0"/>
      <xs:element name="MetricName" type="xs:string" minOccurs="0"/>
      <xs:element name="ProducerID" type="xs:string" minOccurs="0"/>
      <xs:element name="Severity" type="xs:string"/>
      <xs:element name="Message" type="xs:string" minOccurs="0"/>
      <xs:element name="Key1" type="xs:string" minOccurs="0"/>
      <xs:element name="Key2" type="xs:string" minOccurs="0"/>
      <xs:element name="Key3" type="xs:string" minOccurs="0"/>
      <xs:element name="Key4" type="xs:string" minOccurs="0"/>
      <xs:element name="Value" type="xs:string" minOccurs="0"/>
      <xs:element name="TimeStamp" type="xs:dateTime" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:schema>
```

### 3.4 Integrating Enterprise Manager with Oracle BAM

The following sections explain how to use the Data Exchange Connector to integrate Oracle BAM with Enterprise Manager.

#### 3.4.1 Supported Versions

The tested and certified versions of the Oracle BAM server are:

- Oracle BAM server 10gR2 (10.1.2.0.0) and 10gR2 patch sets
- Oracle BAM server 10gR3 (10.1.3.1.0) and 10gR3 patch sets
- Oracle BAM server 11gR1 (11.1.1.1.0) and 11gR1 patch sets
- Oracle BAM server 12c (12.1.4.0.0)
3.4.2 Setting up the Data Flow from Enterprise Manager to Oracle BAM

For successful data flow from Enterprise Manager to Oracle BAM, do the following:

1. Import required artifacts, explained in Importing Oracle BAM Artifacts for an Outbound Session.
2. Update JNDI details, explained in Updating JNDI (for Oracle BAM 10g and 11g only).
3. Run the link plans shown in Table 3–25. (This is only needed for Oracle BAM 10g R3 and previous versions.)

3.4.2.1 Importing Oracle BAM Artifacts for an Outbound Session

Oracle BAM server is not packaged or installed as part of Enterprise Manager. It is assumed that an Oracle BAM instance exists and is up and running. To read and persist the data from Enterprise Manager, certain artifacts should be existing and running. Import the artifacts from the pre-packaged scripts.

To import Oracle BAM artifacts needed for the integration with the Oracle BAM server as a super user, run the following script:

- For Oracle BAM 12c (12.1.4.0.0) or later versions:
  ```bash
  bamcommand -cmd=import -file=emsys_all_11.xml -upgrade 1 -mode append
  ```

- For Oracle BAM 11gR1 (11.1.1.1.0) or later versions:
  ```bash
  ICommand cmd=import file=emsys_all_11.xml
  ```

- For Oracle BAM 10gR3 or older versions:
  ```bash
  ICommand cmd=import file=emsys_all_10.xml
  ```

Both of these files are available at the following location:

$ORACLE_HOME/sysman/bam directory

The export script above creates the following Oracle BAM artifacts:

- EM-BAM Data Objects
- EM-BAM EMS Definitions
- EM-BAM Enterprise Link Plans (only when emsys_all_10.xml is used). See Table 3–25.

EM-BAM Data Objects

Table 3–23 lists the data objects the Import command creates.
### Table 3–23  EM-BAM Data Objects

<table>
<thead>
<tr>
<th>Data Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SYSMAN/EMSSTargets</td>
<td>Contains target metadata information, such as target name and target type.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSMetrics</td>
<td>Contains metric metadata, such as metric name, metric column, and target type.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSAlertsData</td>
<td>Contains the incoming system alerts received from Enterprise Manager. It contains information that includes alert message, alert severity, alert timestamp, and target information on which this alert has occurred.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetSLA</td>
<td>Data object snapshot SLA values.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetSLA Last24Hours</td>
<td>Data object to store the average SLA values for the last 24 hours.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetSLA Last7Days</td>
<td>Data object to store the average SLA values for the last 7 days.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetSLA Last31Days</td>
<td>Data object to store the average SLA values for the last 31 days.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetSLAData</td>
<td>Contains the target SLA information received from Enterprise Manager.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetStatus</td>
<td>Contains target status information, expressed as a percentage.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetStatus Last24Hours</td>
<td>Contains target status information as the average value over 24 hours, expressed as a percentage.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetStatus Last7Days</td>
<td>Contains target status information as the average value over 7 days, expressed as a percentage.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSTargetStatus Last31Days</td>
<td>Contains target status information as the average value over 31 days, expressed as a percentage.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSMetricsData</td>
<td>Data object for RAW metrics.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSMetricsData Last24Hours</td>
<td>Data object to store the average metrics data for the last 24 hours.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSMetricsData Last7Days</td>
<td>Data object to store the average metrics data for the last 7 days.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSMetricsData Last31Days</td>
<td>Data object to store the average metrics data for the last 31 days.</td>
</tr>
<tr>
<td>/SYSMAN/EMSYSSecurityFilter</td>
<td>Acts as the security filter for all other data objects. It contains the session name and users who can access the corresponding session data.</td>
</tr>
</tbody>
</table>

### EM-BAM EMS Definitions

Table 3–24 lists the enterprise message sources the Import command creates.

### Table 3–24  EM-BAM EMS Definitions

<table>
<thead>
<tr>
<th>Data Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMSYSMetricsEMS</td>
<td>Contains the EMS definition for incoming metric metadata listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsTopic.</td>
</tr>
<tr>
<td>EMSYSTargetsEMS</td>
<td>Contains the EMS definition for incoming target metadata listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetsTopic.</td>
</tr>
<tr>
<td>Data Definition</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EMSYSSecurityFilterEMS</td>
<td>Contains the EMS definition for security filter data listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSSecurityFilterTopic.</td>
</tr>
<tr>
<td>EMSYSAlertsDataEMS</td>
<td>Contains the EMS definition for incoming alerts listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSAlertsDataTopic.</td>
</tr>
<tr>
<td>EMSYSMetricsDataEMS</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsDataTopic.</td>
</tr>
<tr>
<td>EMSYSTargetStatusDataEMS</td>
<td>Contains the EMS definition for incoming target status messages metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetStatusTopic.</td>
</tr>
<tr>
<td>EMSYSTargetSLAEMS</td>
<td>Contains the EMS definition for incoming target SLA messages listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetSLATopic.</td>
</tr>
<tr>
<td>EMSYSMetricsEMS-Queue</td>
<td>Contains the EMS definition for incoming metric metadata listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsQueue.</td>
</tr>
<tr>
<td>EMSYSTargetsEMS-Queue</td>
<td>Contains the EMS definition for incoming target metadata listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetsQueue.</td>
</tr>
<tr>
<td>EMSYSSecurityFilterEMS-Queue</td>
<td>Contains the EMS definition for security filter data listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSSecurityFilterQueue.</td>
</tr>
<tr>
<td>EMSYSAlertsDataEMS-Queue</td>
<td>Contains the EMS definition for incoming alerts listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSAlertsDataQueue.</td>
</tr>
<tr>
<td>EMSYSMetricsDataEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsDataQueue.</td>
</tr>
<tr>
<td>EMSYSMetricsDataLast24HoursEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsDataLast24HoursQueue.</td>
</tr>
<tr>
<td>EMSYSMetricsDataLast7DaysEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsDataLast7DaysQueue.</td>
</tr>
<tr>
<td>EMSYSMetricsDataLast31DaysEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsDataLast31DaysQueue.</td>
</tr>
<tr>
<td>EMSYSMetricsDataEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsDataTopic.</td>
</tr>
<tr>
<td>EMSYSMetricsDataLast24HoursEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsDataLast24HoursTopic.</td>
</tr>
<tr>
<td>EMSYSMetricsDataLast7DaysEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsDataLast7DaysTopic.</td>
</tr>
<tr>
<td>Data Definition</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>EMSYMSMetricsDataLast31DaysEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYMSMetricsDataLast31DaysTopic.</td>
</tr>
<tr>
<td>EMSYSTargetSLEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLEMQQueue.</td>
</tr>
<tr>
<td>EMSYSTargetSLALast24HoursEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLALast24HoursQueue.</td>
</tr>
<tr>
<td>EMSYSTargetSLALast7DaysEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLALast7DaysQueue.</td>
</tr>
<tr>
<td>EMSYSTargetSLALast31DaysEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLALast31DaysQueue.</td>
</tr>
<tr>
<td>EMSYSTargetSLEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetSLEMQ.</td>
</tr>
<tr>
<td>EMSYSTargetSLALast24HoursEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLALast24HoursTopic.</td>
</tr>
<tr>
<td>EMSYSTargetSLALast7DaysEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLALast7DaysTopic.</td>
</tr>
<tr>
<td>EMSYSTargetSLALast31DaysEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLALast31DaysTopic.</td>
</tr>
<tr>
<td>EMSYSTargetStatusEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusQueue.</td>
</tr>
<tr>
<td>EMSYSTargetStatusLast24HoursEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusLast24HoursQueue.</td>
</tr>
<tr>
<td>EMSYSTargetStatusLast7DaysEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusLast7DaysQueue.</td>
</tr>
<tr>
<td>EMSYSTargetStatusLast31DaysEMS-Queue</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusLast31DaysQueue.</td>
</tr>
<tr>
<td>EMSYSTargetStatusEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusTopic.</td>
</tr>
<tr>
<td>EMSYSTargetStatusLast24HoursEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusLast24HoursTopic.</td>
</tr>
<tr>
<td>EMSYSTargetStatusLast7DaysEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusLast7DaysTopic.</td>
</tr>
<tr>
<td>EMSYSTargetStatusLast31DaysEMS-Topic</td>
<td>Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetStatusLast31DaysTopic.</td>
</tr>
</tbody>
</table>
3.4.2.2 Updating JNDI (for Oracle BAM 10g and 11g only)
You should update all EMSs described in Table 3–24 to reflect the correct JNDI initial context factory and provider URLs of your JMS servers. To do this:

1. In the Oracle BAM console, click **Architect**.
   
   The Oracle BAM Architect screen appears.

2. Select **Enterprise Message Sources** in the top left drop-down list.
   
   The left pane displays the six Enterprise Message Sources.

3. Click the **Message Source** links and do the following:
   
   a. In the right pane, click **Edit** and modify the JNDI Service Provider URL details from `t3://localhost` to the JNDI Service Provider URL of your Data Hub.
   
   b. Click **Save**.

4. Repeat this for all Message Source objects.

5. Make the following updates depending on the release:
   
   - For 11gR1 or new versions of Oracle BAM:
     
     Enable Global trust or Cross domain trust between the Data hub WebLogic server and the Oracle BAM WebLogic server. Refer to WebLogic documentation for details.
   
   - For 10gR3 or older versions of Oracle BAM:
     
     Update `java.naming.security.principal` to the JMS server password in `jndi.properties` in the `BAM_HOME\BAM\j2re1.4.1_01\lib` directory.

6. Restart all Oracle BAM services.

**EM-BAM Enterprise Link Plans**
Besides the Oracle BAM data objects (Table 3–23) and EMS definitions (Table 3–24), the link plans shown in Table 3–25 are also created when you use `emsys_all_10.xml` to create the artifacts. These are only needed for Oracle BAM Server 10gR3 or older versions. The plans shown in Table 3–25 are created based on the Import command.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMSYSMetricsPlan</td>
<td>Contains the definition to receive, transform, and persist incoming metric metadata messages. This should be running before creating and setting up an outbound session in Enterprise Manager.</td>
</tr>
<tr>
<td>EMSYSTargetsPlan</td>
<td>Contains the definition to receive, transform, and persist incoming target metadata messages. This should be running before creating and setting up an outbound session in Enterprise Manager.</td>
</tr>
</tbody>
</table>
3.4.2.3 Updating the EMS Definitions in Oracle BAM 12c

Do the following to update the definition:

1. For the Enterprise Message source defined under the sysman folder, click the Administrator link on the Oracle BAM Composer console.
2. Expand Enterprise Message Sources in the left panel.
3. Select a Queue or Topic from the left panel, then select Edit.
4. For Queues, select eis/bam/wls/Queue as the Outbound Connection JNDI. Make sure the Queue name is correct. For Topics, select eis/bam/wls/DurableTopic as the Outbound Connection JNDI. Make sure the Topic name and Durable subscriber name are correct.
5. Click Save, then Start.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMSYSSecurityFilterPlan</td>
<td>Contains the definition to receive, transform, and persist incoming security filter messages. This should be running before creating and setting up an outbound session in Enterprise Manager.</td>
</tr>
<tr>
<td>EMSYSAlertsDataPlan</td>
<td>Contains the definition to receive, transform, and persist incoming alert messages. This should be running when the outbound session with at least one selected alert is running in Enterprise Manager.</td>
</tr>
<tr>
<td>EMSYSMetricsDataPlan</td>
<td>Contains definition to receive, transform, and persist incoming metric messages. This should be running when the outbound session with at least one selected metric is running in Enterprise Manager.</td>
</tr>
<tr>
<td>EMSYSTargetStatusDataPlan</td>
<td>Contains definition to receive, transform, and persist incoming target status messages. This should be running when the outbound session with at least one availability selected metric is running in Enterprise Manager.</td>
</tr>
<tr>
<td>EMSYSTargetSLADataPlan</td>
<td>Contains definition to receive, transform, and persist incoming target SLA messages. This should be running when the outbound session with at least one SLA selected metric is running in Enterprise Manager.</td>
</tr>
<tr>
<td>EMSYSAlertsDataRollup</td>
<td>Contains the definition to move the data in EMSYSAlertsData that is more than 24 hours old to EMSYSAlertsData.Archive. You can run this based on demand.</td>
</tr>
<tr>
<td>EMSYSMetricsDataRollup</td>
<td>Contains the definition to move the data in EMSYSMetricsData that is more than 24 hours old to EMSYSMetricsData.Archive. You can run this based on demand.</td>
</tr>
<tr>
<td>EMSYSTargetStatusDataRollup</td>
<td>Contains the definition to move the data in EMSYSTargetStatusData that is more than 24 hours old to EMSYSTargetStatusData.Archive. You can run this based on demand.</td>
</tr>
</tbody>
</table>
6. Repeat this procedure for each Enterprise Message Source that you want to update.

![EMS in Edit Mode](image)

### 3.4.3 Setting up the Data Flow from Oracle BAM to Enterprise Manager

Unlike the Enterprise Manager to Oracle BAM server data transfer, where pre-defined Oracle BAM artifacts, such as Data Objects, EMS, and Plans (when needed) are defined and shipped along with Enterprise Manager, no such artifacts are defined or shipped for the data transfer from Oracle BAM to Enterprise Manager. This is because the Data Object or EMS are unknown.

Consequently, for inbound data transfer from Oracle BAM 11g or later versions, the integrator needs to use Oracle Data Integrator to read the data from the data object and put it on an outbound JMS topic or queue. For Oracle BAM 10g or previous versions, the integrator can use outbound Plans to read data from data objects and put it on an outbound JMS topic or queue. Refer to ODI and Oracle BAM documentation for specific information.

### 3.4.4 End-to-End Flow

After you configure the Oracle BAM server with necessary artifacts and the JMS server with JMS topic or queue names, ensure the following for successful data flow from Enterprise Manager to Oracle BAM:

- Oracle BAM server and Enterprise Manager are up and running.
- JMS server, configured with required JMS topics or queues, is up and running.
- All enterprise plans (only for Oracle BAM 10g or older versions) described in Table 3–25 are running.

    You can start plans manually or schedule them from Design Studio or through an alert.

After ensuring the specifications, proceed with the following:

- Create a data exchange hub and outbound data exchange session instances in Enterprise Manager.

    Always use a normalized message format for sessions that are for the consumption of the Oracle BAM server. Oracle BAM server plans are catered to understand only normalized messages.

- Schedule the outbound data exchange.

    Enterprise Manager sends data as scheduled in the outbound Data Exchange session.

### 3.5 Using an OC4J as a Data Exchange Hub

To use an OC4J as a data exchange hub instead of WebLogic Server, perform the following steps:

1. Copy the client libraries and restart Cloud Control:
   - For a 10.1.3.x OC4J, copy the 10.1.3.x oc4jclient.jar file to:
     ```
     $ORACLE_HOME/middleware/oms/sysman/archives/emgc/deployments/EMGC_DOMAIN/emgc.ear/APP-INF/lib
     ```
   - For a 10.1.2.x OC4J, copy the 10.1.2.x oc4j.jar file to:
     ```
     $ORACLE_HOME//middleware/oms/sysman/archives/emgc/deployments/EMGC_DOMAIN/emgc.ear/APP-INF/lib
     ```

2. Restart Cloud Control after copying the.jar file.

3. Configure the OC4J with the JMS destinations. You can do this with Application Server Control or by manually updating the jms.xml file as follows:

    The following example shows a jms.xml section for an OC4J:

    ```xml
    <topic-connection-factory/>

    <topic>
        name="EMSYSAlertsDataTopic"
        location="jms/EMSYSAlertsDataTopic"
        <description>Topic for alerts data</description>
    </topic>

    <topic>
        name="EMSYSMetricsDataTopic"
        location="jms/EMSYSMetricsDataTopic"
        <description>Topic for metrics data</description>
    </topic>

    <topic>
        name="EMSYSMetricsTopic"
        location="jms/EMSYSMetricsTopic"
        <description>Topic for metrics metadata</description>
    </topic>
    ```
<topic>
    name="EMSYSSecurityFilterTopic"
    location="jms/EMSYSSecurityFilterTopic"
    <description>Topic for security filter</description>
</topic>

<topic>
    name="EMSYSTargetStatusTopic"
    location="jms/EMSYSTargetStatusTopic"
    <description>Topic for target status</description>
</topic>

<topic>
    name="EMSYSTargetSLATopic"
    location="jms/EMSYSTargetSLATopic"
    <description>Topic for target SLA</description>
</topic>

<topic>
    name="EMSYSTargetsTopic"
    location="jms/EMSYSTargetsTopic"
    <description>Topic for targets metadata</description>
</topic>

<queue-connection-factory/>
    location="jms/EMSYSQueueConnectionFactory"

<queue>
    name="EMSYSAlertsDataQueue"
    location="jms/EMSYSAlertsDataQueue"
    <description>Queue for alerts data</description>
</queue>

<queue>
    name="EMSYSMetricsDataQueue"
    location="jms/EMSYSMetricsDataQueue"
    <description>Queue for metrics data</description>
</queue>

<queue>
    name="EMSYSMetricsQueue"
    location="jms/EMSYSMetricsQueue"
    <description>Queue for metrics metadata</description>
</queue>

<queue>
    name="EMSYSSecurityFilterQueue"
    location="jms/EMSYSSecurityFilterQueue"
    <description>Queue for security filter</description>
</queue>

<queue>
    name="EMSYSTargetStatusQueue"
    location="jms/EMSYSTargetStatusQueue"
    <description>Queue for target status</description>
</queue>

<queue>
    name="EMSYSTargetSLAQueue"
3.6 Tips and Troubleshooting Information

The following sections provide various tips and troubleshooting information that might help in resolving various issues you encounter during the data exchange process.

3.6.1 Data Exchange Hub Connection Errors

A data exchange hub connection created in Cloud Control can error out due to authentication issues. These errors can appear in the following places:

- Cloud Control log files
- Cloud Control data exchange pages. For example:

  An error occurred while verifying the Data Exchange hub <hub_name>:
  You are not authorized to access the Data Exchange hub. The <session_name>
  session was not created successfully.

- Data exchange hub logs, such as an OC4J container error from OC4J logs. For example:

  08/02/28 17:37:28 INFO: RMIProto .readConnectionHeader Local ORMI version = 1. 3 different from remote version 1.1 will use 1.1

Follow these steps to resolve the connection problem:

1. Make sure the username/password combination is correct for the data exchange hub. You can check this with client programs, such as JDeveloper or a JMS client.
2. Recreate the Data Exchange hub connection entry in Cloud Control as follows:
   a. Log in to Cloud Control as a super user/administrator.
   b. From Enterprise Manager Cloud Control, click Setup. The setup links appear in the left margin.
   c. Click Data Exchange. The Data Exchange page appears.
   d. In the Data Exchange Hub tab, select the hub connection and click Delete.
   e. Create a new Data Exchange hub. See Creating a Data Exchange Hub.
3.6.2 Notification Methods and Rules

**Important:** The verifications suggested in this section are meant for troubleshooting purposes and not for modification. Updating notification methods or rules can result in undesirable consequences.

**Notification Method**
For each data hub used for an outbound session, a new notification method is created. The name of the method is `hub_nameNotifDevice`, where `hub_name` is the name of the data hub for which the method is created.

**Notification Rules**
For each outbound session (with Alerts selected), a notification rule is created. The name of the rule is `session_nameNotifRule`, where `session_name` is the name of the outbound session for which the rule is created.

To verify that the rule is successfully created:
1. From Enterprise Manager Cloud Control, click the Preferences link.
2. In the left pane under Notification, click Rules.
   The Notification Rules page appears.
3. Click on the corresponding Rule and make sure the selected targets and metrics are correct.

3.6.3 Data Flow Tips

- Ensure that the following JNDI details are correctly entered for the Data Hub you use:
  - JNDI URL
  - Username
  - Password

- For an inbound session setup, do not provide JNDI credentials for JMS in the data source definitions.
  If the JMS topic or queue is secured by providing authentication details, provide the username and password. If not, leave the fields blank.

- Ensure that the JMS server is up and running and configured with the required JMS topic and queue names.

- Ensure that either an inbound or outbound session is scheduled and is running.

- For an inbound session data source, to ensure that the topic details are correctly entered, click Test Connection.

- Using the same topic or queue name for two active inbound sessions could result in corrupted data.

- Frequency for an inbound session should either be synchronized with or relative to the frequency at which the external system sends data.

  For example, setting the inbound session frequency to 2 minutes is ineffective if the external system sends data only once in 10 minutes.
For an outbound session, ensure that Receiver or EL Plans (in the case of Oracle BAM) are running.

To improve efficiency, outbound session schedule frequency should be relative or synchronized with the frequencies at which Enterprise Manager collects the metrics defined in the session.

For example, in Enterprise Manager, if the collection frequency for metrics defined in the outbound session is 5 minutes, setting a lesser outbound frequency (one minute, for instance) is ineffective, as the possibility of new data is remote. In such cases, setting the outbound frequency to 5 or 10 minutes would be effective.

Note that only new metric values (if any) are forwarded.

Do not schedule two or more outbound sessions with different message formats at the same time.

Unless a lower frequency level is absolutely required, always set higher frequency intervals. This helps to reduce the Enterprise Manager/JMS server load.

### 3.6.4 Log Files

Always check the log if data could not be received or if the status is Failure. To check logs:

1. From Enterprise Manager Cloud Control, click the **Jobs** tab.
2. Click **Advanced Search**.
3. In **Target Type**, select **Targetless** from the drop-down list.
4. In **Status**, select **All** from the drop-down list.
5. Click **Go**.
6. Click the Job you want to verify.
   - For an inbound session, the name of the job is `<Session Name>ISJOB`.
   - For an outbound session, the name of the job is `<Session Name>OSJOB`.
7. Click the Status value link; for instance, **Succeeded** or **Failed**.
8. Click **Step**.
   - For inbound sessions, the step name is `receiveMetricDataViaJms`.
   - For outbound sessions, the step name is `sendMetricDataViaJms`.
9. Note the Step ID Value and search logs (typically located in the directory `$ORACLE_HOME/sysman/log`) based on the ID in the log directory using the following command:

   "grep -i "JobWorker stepID" *.trc"

   **Note:** The default system log directory is `$ORACLE_HOME/sysman/log`

10. Check for the following:
    - Exceptions or errors
    - `emoms.log` (in the same directory) for any other exception for the same timestamp
11. Change the log level in `emomslogging.properties` to `DEBUG` and restart Enterprise Manager for more detailed debugging information.

### 3.6.5 End-to-End Flow Sample Demonstrations

For the convenience of integrators, Oracle provides sample demonstrations detailing the end-to-end data flow. To access the demonstrations, go to the following directory:

```
$ORACLE_HOME/sysman/bam
```

Instructions for an outbound session sample are provided in the file `outboundsession_sample_readme.txt`. You can create the report required for the demonstration by importing the file `outboundsession_report.xml` as detailed in the readme file.

Instructions for an inbound session sample are provided in the file `inboundsession_sample_readme.txt`. You can create the artifacts required for the demonstration by importing the file `inboundsession.xml` as detailed in the readme file.

### 3.7 Suggested Reading

The following list provides online resources that can help you effectively use all associated technologies involved in the data exchange process.

- **Oracle Business Activity Monitoring:**
  

- **Java Message Service**
  
  - **Java Message Service Concepts:**
    

  - **Java Message Service Specification:**
    
    [https://java.net/projects/jms-spec/pages/Home](https://java.net/projects/jms-spec/pages/Home)
This chapter provides tabular reference information for connectors. The following sections provide reference tables for the following categories:

- Request Attributes
- Response File Properties for the Windows Platform
-Queryable Properties
- Complex Response Properties
- Status Codes

This chapter also provides information about the response file properties that the Create RAC and Add Node jobs generate for the Windows platform.

### 4.1 Request Attributes

The tables in this section provide query paths, descriptions, and data types for the following property types:

- setModel Request
- Request Header
- Create RAC
- Add Node
- Delete Node

Table 4–1 provides path types, descriptions, and data types for setModel request elements.

<table>
<thead>
<tr>
<th>Path Type</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;RequestHeader/&gt; Request header. See Table 4–2. Complex Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Credential&gt; Credential for Enterprise Manager. Complex Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Name&gt;sysman&lt;/Name&gt; User name. String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Password&gt;welcome1&lt;/Password&gt; Password. String</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4–1 (Cont.) setModel Request Elements

<table>
<thead>
<tr>
<th>Path Type</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;/Credential&gt;</td>
<td>End of Credential.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;AggregateTarget&gt;</td>
<td>EMMModel should contain two aggregate targets: one of type cluster, and one of type rac_database. This aggregate target is of type cluster. It contains information about the cluster.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Name&gt;CRS30&lt;/Name&gt;</td>
<td>Name of the cluster.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Type&gt;cluster&lt;/Type&gt;</td>
<td>Aggregate target type.</td>
<td>String (enumeration: &quot;cluster&quot; &quot;rac_database&quot;)</td>
</tr>
<tr>
<td>&lt;Target&gt;</td>
<td>Number of targets in the cluster aggregate target should be the same as the number of hosts in the cluster. Each target element contains information about a host in the cluster.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Name&gt;bxj30&lt;/Name&gt;</td>
<td>Name of the host.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Type&gt;host&lt;/Type&gt;</td>
<td>Target type. In the cluster aggregate target, it should be set to &quot;host.&quot;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Host&gt;bxj30&lt;/Host&gt;</td>
<td>Name of the host.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Credential&gt;</td>
<td>Credential of the host.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Name&gt;oracle&lt;/Name&gt;</td>
<td>User name.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Password&gt;welcome1</td>
<td>Password.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;/Password&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;/Credential&gt;</td>
<td>End of Credential.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Property/&gt;</td>
<td>Property of the target. See the corresponding table in this section for the properties of the &quot;host&quot; target type.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;/Target&gt;</td>
<td>End of Target.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Property/&gt;</td>
<td>Property of the cluster aggregate target. See the corresponding table in this section for the properties of the &quot;cluster&quot; aggregate target type.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;/AggregateTarget&gt;</td>
<td>End of AggregateTarget.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;AggregateTarget&gt;</td>
<td>EMMModel should contain two aggregate targets: one of type cluster and one of type rac_database. This aggregate target is of type rac_database. It contains information about the RAC.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Name&gt;RAC30&lt;/Name&gt;</td>
<td>Name of the RAC database.</td>
<td>String (length &lt;=8)</td>
</tr>
<tr>
<td>&lt;Type&gt;rac_database&lt;/Type&gt;</td>
<td>Aggregate target type.</td>
<td>String (enumeration: &quot;cluster&quot; or &quot;rac_database&quot;)</td>
</tr>
<tr>
<td>&lt;Storage&gt;</td>
<td>Storage element contains storage information for the RAC. This element can be omitted for the Add Node job request if the storage type is not ASM.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;Type&gt;ASM&lt;/Type&gt;</td>
<td>Type of storage.</td>
<td>String (enumeration: &quot;CFS&quot; or &quot;ASM.&quot; &quot;RAW&quot; is not supported.)</td>
</tr>
<tr>
<td>&lt;Property/&gt;</td>
<td>Properties for storage. See the corresponding table in this section for the storage properties.</td>
<td>Complex Type</td>
</tr>
<tr>
<td>&lt;/Storage&gt;</td>
<td>End of Storage.</td>
<td>Complex Type</td>
</tr>
</tbody>
</table>
Table 4–2 provides path types, descriptions, and data types for request header elements.

### Table 4–2 Request Header Elements

<table>
<thead>
<tr>
<th>Path Type</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;RequestID/&gt;</td>
<td>Uniquely identifies the request. This is mainly used by the client. Enterprise Manager currently does not track this ID.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Source/&gt;</td>
<td>Request source, which is the request operating system.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Destination/&gt;</td>
<td>Destination should be Enterprise Manager in this release.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;RequestProperty&gt;</td>
<td>Platform is an optional property. Specify either Linux or Windows. If you do not specify a platform, the default is Linux. The platform is only relevant in provisioning use cases.</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 4–3 provides target types, properties, descriptions, and data types for Create RAC.
<table>
<thead>
<tr>
<th>Target Type</th>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Target&gt;</td>
<td>CRS_HOME</td>
<td>Oracle Clusterware home directory. This property must be the same for all hosts in the cluster.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Type&gt;host&lt;/Type&gt;</td>
<td>ORACLE_HOME_NAME</td>
<td>Oracle Clusterware home name. This is an optional property. The default value is the cluster aggregate target name.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;/Target&gt;</td>
<td>publicNode</td>
<td>Public node name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>privateNode</td>
<td>Private node name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>vipNode</td>
<td>Virtual node name.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Target&gt;</td>
<td>ORACLE_HOME</td>
<td>Database home directory. This property must be the same for all database instances in the RAC database.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Type&gt;oracle_database&lt;/Type&gt;</td>
<td>ORACLE_HOME_NAME</td>
<td>Database home name. This property is optional. The default value is the rac_database aggregate target name.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;/Target&gt;</td>
<td>db_username</td>
<td>Database user for setting monitoring credentials. It should always be SYS in this release.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>db_password</td>
<td>Password of the database user. It is the password for SYS and SYSTEM in this release.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>oms_username</td>
<td>Oracle Management Services host operating system user name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>oms_password</td>
<td>Oracle Management Services host operating system password.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Storage&gt;</td>
<td>diskGroupName</td>
<td>ASM disk group name.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Type&gt;ASM&lt;/Type&gt;</td>
<td>diskList</td>
<td>Disk device list. Use a comma (,) as a separator.</td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td>&lt;/Storage&gt;</td>
<td>diskString</td>
<td>Search paths for ASM disks.</td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td></td>
<td>redundancy</td>
<td>Redundancy level.</td>
<td>String (enumeration: NORMAL, HIGH, or EXTERNAL)</td>
</tr>
<tr>
<td></td>
<td>asmPassword</td>
<td>ASM SYSDBA password.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Storage&gt;</td>
<td>datafileDestination</td>
<td>Data file directory.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Type&gt;CFS&lt;/Type&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;/Storage&gt;</td>
<td>softwareImageName</td>
<td>Name of the software library image for the CRS home. This property is optional. The default value is the latest active software library image of type &quot;Oracle Clusterware Clone.&quot;</td>
<td>String</td>
</tr>
</tbody>
</table>
### Table 4–3  (Cont.) Create RAC Properties

<table>
<thead>
<tr>
<th>Target Type</th>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s_ocrpartitionlocation</td>
<td>OCR location. Use the comma (,) as a separator. This property is only for the Linux platform.</td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td></td>
<td>s_votingdisklocation</td>
<td>Voting disk location. Use the comma (,) as a separator. This property is only for the Linux platform.</td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td></td>
<td>RESPONSEFILE_VERSION</td>
<td>Response file version. This property is optional. The default value is 2.2.1.0.0. This property is only for the Windows platform.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>sl_OHPartitionsAndSpace_valueFromDlg</td>
<td>This property specifies the split-up of disk partitions for OCR/Vdisk locations. This property is only for the Windows platform.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>re1_PrivIntrList</td>
<td>This property specifies the interconnects to use. This property is only for the Windows platform.</td>
<td>String</td>
</tr>
</tbody>
</table>

```xml
<AggregateTarget>
  <Type>rac_database</Type>
</AggregateTarget>
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateName</td>
<td>Database template name. This property is optional. The default value is General_Purpose.dbc</td>
<td>String (file name without path)</td>
</tr>
<tr>
<td>gdbName</td>
<td>Global database name. This property is optional. The default value is the rac_database aggregate target name.</td>
<td>String (length &lt;=8)</td>
</tr>
<tr>
<td>sid</td>
<td>Database instance name. This should be the same as gdbName in this release. This property is optional. The default value is the rac_database aggregate target name.</td>
<td>String (length &lt;=8)</td>
</tr>
<tr>
<td>characterSet</td>
<td>Character set. See the Database Globalization Support guide for details. This property is optional. The default value is UTF8.</td>
<td>String</td>
</tr>
</tbody>
</table>
Table 4–3 (Cont.) Create RAC Properties

<table>
<thead>
<tr>
<th>Target Type</th>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>&lt;Storage&gt;</em></td>
<td>nationalCharacterSet</td>
<td>National character set. See the Database Globalization Support guide for details. This property is optional. The default value is UTF8.</td>
<td>String</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>initParams</td>
<td>Raw strings for additional input. For example: <code>nls_territory=japan, nls_language=japanese</code></td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>softwareImageName</td>
<td>Name of the software library image for the database home. This property is optional. The default value is: <code>nls_lang=american,nls_territory=american</code></td>
<td>String</td>
</tr>
</tbody>
</table>

Table 4–4 provides target types, properties, descriptions, and data types for Add Node for a RAC aggregate target.

Table 4–4 Add Node Properties (Storage and Aggregate Target)

<table>
<thead>
<tr>
<th>Target Type</th>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>&lt;Storage&gt;</em></td>
<td>asmPassword</td>
<td>ASM SYSDBA password.</td>
<td>String</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>socrpartitionlocation</td>
<td>OCR location. Use the comma (,) as a separator. This property is only for the Linux platform. Set this value to be the same as the Create RAC job.</td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>svotingdisklocation</td>
<td>Voting disk location. Use the comma (,) as a separator. This property is only for the Linux platform. Set this value to be the same as the Create RAC job.</td>
<td>String (no space allowed)</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>RESPONSEFILE_VERSION</td>
<td>Response file version. This property is optional. The default value is 2.2.1.0.0. This property is only for the Windows platform. Set this value to be the same as the Create RAC job.</td>
<td>String</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>sl_OHPartitionsAndSpace_valueFromDlg</td>
<td>This property specifies the split-up of disk partitions for OCR/Vdisk locations. This property is only for the Windows platform. Set this value to be the same as the Create RAC job.</td>
<td>String</td>
</tr>
<tr>
<td><em>&lt;AggregateTarget&gt;</em></td>
<td>ret_PrivIntrList</td>
<td>This property specifies the interconnects to use. This property is only for the Windows platform. Set this value to be the same as the Create RAC job.</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 4–5 provides target types, properties, descriptions, and data types for Add Node for a new node.
**Table 4-5  Add Node Properties (New Node)**

<table>
<thead>
<tr>
<th>Target Type</th>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Target&gt;</td>
<td>&lt;Type&gt;host&lt;/Type&gt;</td>
<td>CRS_HOME Oracle Clusterware home directory. This property must be the same for all hosts in the cluster.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORACLE_HOME_NAME Oracle Clusterware home name. This property is optional. The default value is cluster aggregate target name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>publicNode</td>
<td>Public node name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>privateNode</td>
<td>Private node name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>vipNode</td>
<td>Virtual node name.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Target&gt;</td>
<td>&lt;Type&gt;oracle_database&lt;/Type&gt;</td>
<td>ORACLE_HOME Database home directory. This property must be the same for all database instances in the RAC database.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ORACLE_HOME_NAME Database home name. This property is optional. The default value is rac_database aggregate target name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>db_username</td>
<td>Database user name that has a SYSDBA role.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>db_password</td>
<td>Password of the user above.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>oms_username</td>
<td>Oracle Management Services host operating system user name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>oms_password</td>
<td>Oracle Management Services host operating system password.</td>
<td>String</td>
</tr>
</tbody>
</table>

**Table 4-6 provides target types, properties, descriptions, and data types for Add Node for any existing node.**

**Table 4-6  Add Node Properties (Any Existing Node)**

<table>
<thead>
<tr>
<th>Target Type</th>
<th>Property</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Target&gt;</td>
<td>&lt;Type&gt;host&lt;/Type&gt;</td>
<td>publicNode Public node name.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRS_HOME Oracle Clusterware home directory. This property must be the same for all hosts in the cluster.</td>
<td>String</td>
</tr>
<tr>
<td>&lt;Target&gt;</td>
<td>&lt;Type&gt;oracle_database&lt;/Type&gt;</td>
<td>ORACLE_HOME Database home directories. This property must be the same for all database instances in the RAC database.</td>
<td>String</td>
</tr>
</tbody>
</table>

**Table 4-7 provides target types, properties, descriptions, and data types for Delete Node for remaining nodes.**
4.2 Response File Properties for the Windows Platform

The following properties are required to generate the response file for the Create RAC and Add Node jobs on the Windows platform:

- sl_OHPartitionsAndSpace_valueFromDlg
- ret_PrivIntrList

The following sections describe each response file property.

4.2.1 sl_OHPartitionsAndSpace_valueFromDlg Property

This property specifies the splitting up of disk partitions for OCR/Vdisk locations. It consists of the following six values for each location:

- Disk no.
- Partition no.
- Partition Size (MB)
- Format Type
  - 0: None/RAW
  - 1: CFS for data
  - 2: CFS for software
- Drive Letter
  - N/A: RAW
  - "Available" drive letter: CFS
- Usage Type
– 0: Data/software use ONLY
– 1: OCR primary ONLY
– 2: Voting disk ONLY
– 3: OCR primary and voting disk on the same CFS partition
– 4: OCR mirror only
– 5: OCR mirror and voting disk on the same CFS partition

**Example 1**

Given the following scenario:

■ OCR and the Voting Disk are on Partition-2 of Disk-1 (Partition-2 has size 10002 MB).
■ The partition is CFS-formatted.
■ Both OCR and the Voting Disk reside on the same partition.
■ The drive letter for the partition is G:.
■ There is only data storage and no software storage.

You would specify sl_OHPartitionsAndSpace_valueFromDlg as follows:

```xml
<Property>
  <Name>sl_OHPartitionsAndSpace_valueFromDlg</Name>
  <Value>"1", "2", "10002", "1", "G:", "3"</Value>
</Property>
```

**Example 2**

Given the following scenario:

■ OCR and the Voting Disk reside on different partitions.
■ OCR is on Partition-1 of Disk-3, which has a size of 486 MB and is RAW-formatted.
■ The Voting Disk is on Partition-1 of Disk-4, which has a size of 486 MB and is RAW-formatted.

You would specify sl_OHPartitionsAndSpace_valueFromDlg as follows:

```xml
<Property>
  <Name>sl_OHPartitionsAndSpace_valueFromDlg</Name>
  <Value>"3", "1", "486", "0", "N/A", "1", "1", "486", "1", "486", "0", "N/A", "2"</Value>
</Property>
```

**4.2.2 ret_PrivIntrList Property**

This property specifies the interconnects to use. You should specify entries in ret_PrivIntrList as a comma-separated list of interfaces. Each entry should be a colon-separated string with three fields. You should specify the fields as follows:

■ The first field should be the interface name.
■ The second field should be the subnet IP of the interface.
■ The third field should indicate how Oracle Clusterware should use the interface: as a public interface, private interface, or whether it should not be used at all by the clusterware. This field should be specified as a number — 1, 2, or 3. These numbers represent the following values:
Queryable Properties

- 1: Public
- 2: Private
- 3: Do not use

Example

Given the following scenario:

■ One "Local Area Connection" public interconnect is to be used.
■ One "Local Area Connection2" private interconnect is to be used.

You would specify `ret_PrivIntrList` as follows:

```xml
<Property>
  <Name>ret_PrivIntrList</Name>
  <Value>"Local Area Connection:123.45.67.0:1","Local Area Connection2:123.45.89.0:2"
</Property>
```

4.3 Queryable Properties

The tables in this section provide property names, descriptions, and data types for the following types of queryable properties:

- General Target
- Oracle Database
- Oracle Listener
- Host Target
- Cluster
- Cluster Database
- Oracle Enterprise Manager Agent
- Oracle Enterprise Manager Repository Target
- Job

Table 4–8 provides property names, descriptions, and data types for general target queryable properties.

### Table 4–8 General Target Properties

<table>
<thead>
<tr>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:status)</td>
<td>Integer status of the Enterprise Manager target instance. (See Table 4–20.)</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:monitoring agent)</td>
<td>Enterprise Manager target instance name (of type <code>oracle_emd</code>) of the Agent monitoring the Enterprise Manager target instance.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:homepage)</td>
<td>Enterprise Manager Console home page URI (the path portion of the URL, as in <code>/em/console?...</code>) of the Enterprise Manager target instance.</td>
<td>URL</td>
</tr>
<tr>
<td>Property(Name:version)</td>
<td>Version of the Enterprise Manager target instance.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:oracle home)</td>
<td>Oracle home of the Enterprise Manager target instance. The form of the directory path (path separator) is not further specified here.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:critical alerts)</td>
<td>Number of critical alerts against the Enterprise Manager target instance.</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:warning alerts)</td>
<td>Number of warning alerts against the Enterprise Manager target instance.</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:critical policy violations)</td>
<td>Number of critical policy violations against the Enterprise Manager target instance.</td>
<td>Integer</td>
</tr>
</tbody>
</table>
Queryable Properties

<table>
<thead>
<tr>
<th>Table 4–9 Oracle Database Properties</th>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:instance name)</td>
<td>Instance name of the database instance.</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Property(Name:listener)</td>
<td>Listener Enterprise Manager target instance name (of type oracle_listener) of the listener for the database instance.</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Property(Name:is archiving)</td>
<td>The value is 1 if high availability archiving is on for the Oracle database instance. Otherwise, the value is 0.</td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>Property(Name:is flashback logging)</td>
<td>The value is 1 if high availability flashback logging is on for the Oracle database instance. Otherwise, the value is 0.</td>
<td>Integer</td>
<td></td>
</tr>
</tbody>
</table>

Table 4–10 provides property names, descriptions, and data types for Oracle listener properties. The target type for the Oracle listener is oracle_listener.

<table>
<thead>
<tr>
<th>Table 4–10 Oracle Listener Properties</th>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:alias)</td>
<td>Alias of the Oracle Listener instance.</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Property(Name:net address)</td>
<td>Net address of the Oracle Listener instance.</td>
<td>URI</td>
<td></td>
</tr>
<tr>
<td>Property(Name:listener.ora location)</td>
<td>File directory location of the listener.ora file of the Oracle Listener instance. The form of the directory path (path separator) is not further specified here.</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Property(Name:start name)</td>
<td>Start time of the Oracle Listener instance. The form of this time stamp is not further specified here.</td>
<td>Time</td>
<td></td>
</tr>
</tbody>
</table>

Table 4–11 provides property names, descriptions, and data types for host target properties. The target type for the Host is host.

To get the targets within the domain of a cluster, first request the cluster hosts with the "Target" sub-element. Then get all the targets and filter the list by the hosts in the cluster hosts list.

<table>
<thead>
<tr>
<th>Table 4–11 Host Target Properties</th>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:cluster)</td>
<td>Enterprise Manager target instance name (of type cluster) of the cluster for this host instance.</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Property(Name:cpu utilization)</td>
<td>CPU utilization as a real number between 0 and 1 (inclusive) of the host.</td>
<td>Number</td>
<td></td>
</tr>
<tr>
<td>Property(Name:memory utilization)</td>
<td>Memory utilization as a real number between 0 and 1 (inclusive) of the host.</td>
<td>Number</td>
<td></td>
</tr>
<tr>
<td>Property(Name:total io rate)</td>
<td>Total I/O per second.</td>
<td>Number</td>
<td></td>
</tr>
</tbody>
</table>
Queryable Properties

Table 4–12 provides property names, descriptions, and data types for cluster properties. The target type for Oracle Clusterware is `cluster`.

<table>
<thead>
<tr>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:version)</td>
<td>Clusterware version. Note that this property definition just redefines the same property defined for the general target mappings.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:cluster database)</td>
<td>Cluster databases (of type <code>rac_database</code>).</td>
<td>String</td>
</tr>
<tr>
<td>Target</td>
<td>Cluster hosts (of type host).</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 4–13 provides property names, descriptions, and data types for cluster database properties. The target type for the Oracle cluster database is `rac_database`.

<table>
<thead>
<tr>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:cluster)</td>
<td>Enterprise Manager target instance name (of type cluster) of the cluster for this database instance.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:database name)</td>
<td>Database instance name.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:is archiving)</td>
<td>Value is 1 if high availability archiving is on for the cluster database. Otherwise, the value is 0.</td>
<td>Integer</td>
</tr>
<tr>
<td>Target</td>
<td>Cluster database instance of type <code>oracle_database</code>.</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 4–14 provides property names, descriptions, and data types for Oracle Enterprise Manager Agent properties. The target type for the Oracle Management Agent is `oracle_emd`.

<table>
<thead>
<tr>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:management service)</td>
<td>OMS that the Enterprise Manager Agent instance uploads to.</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 4–15 provides property names, descriptions, and data types for Oracle Enterprise Manager Repository target properties. The target type for the Oracle Management Repository is `oracle_emrep`.

<table>
<thead>
<tr>
<th>Query Path</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(Name:agent count)</td>
<td>Number of Agents for this repository instance.</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:target count)</td>
<td>Number of targets for this repository instance.</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:administrator count)</td>
<td>Number of administrators for this repository instance.</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:session count)</td>
<td>Number of active Oracle management services repository sessions for this repository instance.</td>
<td>Integer</td>
</tr>
<tr>
<td>Property(Name:Integer)</td>
<td>Enterprise Manager database target instance(s) of the database(s) for this repository instance. This property is expanded into complex property elements in the response as described in Table 4–18. They are keyed by the &quot;name&quot; and &quot;value&quot; sub-properties.</td>
<td>String</td>
</tr>
<tr>
<td>Property(Name:tablespace)</td>
<td>Expands to the tablespaces used in the database for this repository instance.</td>
<td>String</td>
</tr>
</tbody>
</table>
4.4 Complex Response Properties

The tables in this section provide property names, descriptions, and data types for the following complex properties returned in the response model to the query requests:

- Oracle Management Service (OMS)
- Database instance

Table 4–17 provides property names, descriptions, and data types for the Oracle Management Service. The type of the complex property is OMS.

Table 4–18 provides property names, descriptions, and data types for the database instance. The type of the complex property is OMS.
4.5 Status Codes

The following tables provide status codes and descriptions for the following status types:

- Enterprise Manager
- Jobs

Table 4–19 describes the status codes for Enterprise Manager. You can use online help for a detailed description of Enterprise Manager target statuses. Enter Target Status as the keywords to search in online help, then select the topic About the Status Icons.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Target down</td>
</tr>
<tr>
<td>1</td>
<td>Target up</td>
</tr>
<tr>
<td>2</td>
<td>Metric error</td>
</tr>
<tr>
<td>3</td>
<td>Agent down</td>
</tr>
<tr>
<td>4</td>
<td>Unreachable</td>
</tr>
<tr>
<td>5</td>
<td>Blackout</td>
</tr>
<tr>
<td>6</td>
<td>Pending/unknown</td>
</tr>
</tbody>
</table>

Table 4–20 describes the status codes for jobs. You can use online help for a detailed description of Enterprise Manager job statuses. Enter Job Status as the keywords to search in online help, then select the topic About Job Status.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCHEDULED — The execution is in a scheduled state.</td>
</tr>
<tr>
<td>2</td>
<td>RUNNING — The execution is running.</td>
</tr>
<tr>
<td>3</td>
<td>INITIALIZATION ERROR — The execution encountered an error and the remote process did not run.</td>
</tr>
<tr>
<td>4</td>
<td>FAILED — The execution failed.</td>
</tr>
<tr>
<td>5</td>
<td>SUCCEEDED — The execution succeeded.</td>
</tr>
<tr>
<td>6</td>
<td>SUSPENDED BY USER — A user suspended the execution.</td>
</tr>
<tr>
<td>7</td>
<td>SUSPENDED ON AGENT UNREACHABLE — The execution was suspended because the Agent was unreachable.</td>
</tr>
<tr>
<td>8</td>
<td>STOPPED — A user stopped the execution.</td>
</tr>
<tr>
<td>9</td>
<td>SUSPENDED ON LOCK — The execution is waiting for a lock on a shared resource.</td>
</tr>
<tr>
<td>10</td>
<td>SUSPENDED ON EVENT — The execution is waiting for an event to occur (usually for an Agent to bounce).</td>
</tr>
<tr>
<td>11</td>
<td>SUSPENDED ON BLACKOUT — The execution is suspended on a blackout.</td>
</tr>
<tr>
<td>12</td>
<td>STOP PENDING — The execution is in Stop Pending status waiting for some running steps to finish.</td>
</tr>
</tbody>
</table>
Table 4–20 (Cont.) Job Status Codes

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>SUSPEND PENDING — The execution is in Suspend Pending status waiting for some running steps to finish.</td>
</tr>
<tr>
<td>14</td>
<td>Inactive (internal state).</td>
</tr>
<tr>
<td>15</td>
<td>Queued (internal state).</td>
</tr>
<tr>
<td>16</td>
<td>Failed retried (internal state).</td>
</tr>
<tr>
<td>18</td>
<td>SKIPPED — The execution was skipped and could not run, because the previous run of the job required too much time, or the Agent was unavailable for too long a period of time.</td>
</tr>
<tr>
<td>20</td>
<td>REASSIGNED — The execution is suspended because the original owner of the job was deleted and the job is not assigned to a new owner. The new owner must explicitly resume the job from the console.</td>
</tr>
<tr>
<td>21</td>
<td>SUSPENDED ON MISSING CREDENTIALS — The execution is suspended because some of the credentials needed for the job are not set.</td>
</tr>
</tbody>
</table>
This appendix provides the following sample implementations for Remedy 7.6 Help Desk ticketing connectors:

### Sample Implementation Files (xml and .xsl):
- connectorDeploy.xml
- getTicket_request.xml
- getTicket_response.xsl
- createTicket_response.xsl
- publishTicket_request.xsl
- Remedy_DefaultCategory.xsl
- Remedy_DefaultCategory_AutoClose.xsl
- Remedy_DefaultCategory_AutoResolve.xsl

### Sample Schema Files (.xsd):
- connectorDeploy.xsd
- EMEvent.xsd
- EMIncident.xsd
- externalEvent.xsd
- createTicket_response.xsd
- getTicket_request.xsd
- getTicket_response.xsd
- publishTicket.xsd

---

**Example A–1 connectorDeploy.xml**

```xml
<?xml version='1.0' encoding='UTF-8' ?>
<ManagementConnector xmlns="http://xmlns.oracle.com/sysman/connector">
  <Name>Remedy Service Desk 7.6 Connector</Name>
  <Version>12.1.0.2.0</Version>
  <EMCompatibleVersion>12.1.0.1.0</EMCompatibleVersion>
  <Description>Remedy Service Desk 7.6.04 Integration with Enterprise Manager</Description>
  <Category>TicketingConnector</Category>
  <SOAPHeaderAuthentication>
    <Username required="true">
      <VariableName>USERNAME</VariableName>
      <DisplayName>Remedy Username</DisplayName>
    </Username>
    <Password>
      <VariableName>PASSWORD</VariableName>
      <DisplayName>Remedy Password</DisplayName>
    </Password>
    <AuthVariable>
      <VariableName>AUTHENTICATION</VariableName>
      <DisplayName>Authentication</DisplayName>
    </AuthVariable>
    <AuthVariable>
      <VariableName>LOCALE</VariableName>
      <DisplayName>Locale</DisplayName>
    </AuthVariable>
  </SOAPHeaderAuthentication>
</ManagementConnector>
```
<AuthVariable>
  <VariableName>TIMEZONE</VariableName>
  <DisplayName>Timezone</DisplayName>
</AuthVariable>

<SOAPHeader>
  <![CDATA[
    <urn:AuthenticationInfo xmlns:urn="urn:HelpDesk_Submit_Service">
      <urn:userName>$USERNAME$</urn:userName>
      <urn:password>$PASSWORD$</urn:password>
      <urn:authentication>$AUTHENTICATION$</urn:authentication>
      <urn:locale>$LOCALE$</urn:locale>
      <urn:timeZone>$TIMEZONE$</urn:timeZone>
    </urn:AuthenticationInfo>
  ]]>}
</SOAPHeader>

<ConnectivityTestVariable>
  <VariableName>TICKET_ID</VariableName>
  <DisplayName>Ticket ID</DisplayName>
</ConnectivityTestVariable>

<Service>
  <Method>createTicket</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[midtier_server]/arsys/services/ARService?server=[servername]&webService=HPD_IncidentInterface_Create_WS]]>
  </WebServiceEndpoint>
</Service>

<Service>
  <Method>updateTicket</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[midtier_server]/arsys/services/ARService?server=[servername]&webService=HPD_IncidentInterface_Custom_WS]]>
  </WebServiceEndpoint>
</Service>

<Service>
  <Method>getTicket</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[midtier_server]/arsys/services/ARService?server=[servername]&webService=HPD_IncidentInterface_Custom_WS]]>
  </WebServiceEndpoint>
</Service>

<ExternalURL>
  <Pattern>
    <![CDATA[http://$WEB_SERVER$/arsys/forms/$ARSERVER_NAME$/$FORM_NAME$/?qual=%27Incident%20Number*%27=%22@Incident_Number@%22]]>
  </Pattern>
  <ConfigVariable required="true">
    <VariableName>WEB_SERVER</VariableName>
    <DisplayName>Web Server</DisplayName>
  </ConfigVariable>
  <ConfigVariable required="true">
    <VariableName>FORM_NAME</VariableName>
    <DisplayName>HelpDesk Case Form Name</DisplayName>
  </ConfigVariable>
  <ConfigVariable required="true">
    <VariableName>ARSERVER_NAME</VariableName>
    <DisplayName>ARServer Name</DisplayName>
  </ConfigVariable>
Ticketing Connector Samples

Example A–2  getTicket_request.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<urn:HelpDesk_Query_Service xmlns:urn="urn:HPD_IncidentInterface_Custom_WS">
  <urn:Incident_Number>@TicketId@</urn:Incident_Number>
</urn:HelpDesk_Query_Service>
```
Example A–3  getTicket_response.xsl

<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:urn="urn:HPD_IncidentInterface_Custom_WS"
  xmlns="http://xmlns.oracle.com/sysman/connector/tt"
  targetNamespace="http://xmlns.oracle.com/sysman/connector/tt"
  elementFormDefault="qualified">
  <xsl:template match="urn:HelpDesk_Query_ServiceResponse">
    <getTicketResponse>
      <TicketId><xsl:value-of select="urn:Incident_Number/text()"/></TicketId>
    </getTicketResponse>
  </xsl:template>
</xsl:transform>

Example A–4  createTicket_response.xsl

<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:urn="urn:HPD_IncidentInterface_Create_WS"
  xmlns="http://xmlns.oracle.com/sysman/connector">
  <xsl:template match="urn:HelpDesk_Submit_ServiceResponse">
    <CreateTicketResponse>
      <TicketId>
        <xsl:value-of select="urn:Incident_Number"/>
      </TicketId>
      <InstanceVariable>
        <VariableName>Incident_Number</VariableName>
        <VariableValue>
          <xsl:value-of select="urn:Incident_Number"/>
        </VariableValue>
      </InstanceVariable>
    </CreateTicketResponse>
  </xsl:template>
</xsl:transform>

Example A–5  publishTicket_request.xsl

<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:a="http://xmlns.oracle.com/sysman/connector"
  targetNamespace="http://xmlns.oracle.com/sysman/connector"
  elementFormDefault="qualified">
  <xsl:template match="a:InboundData">
    <InboundData>
      <Operation><xsl:value-of select="a:Operation"/></Operation>
      <PropertyList>
        <ticket_guid><xsl:value-of select="a:ticket_guid"/></ticket_guid>
      </PropertyList>
      <status>
        <xsl:choose>
          <xsl:when test="[a:PropertyList/a:status/text() = '0']">New</xsl:when>
          <xsl:when test="[a:PropertyList/a:status/text() = '1']">Assigned</xsl:when>
          <xsl:when test="[a:PropertyList/a:status/text() = '2']">In Progress</xsl:when>
        </xsl:choose>
      </status>
    </InboundData>
  </xsl:template>
</xsl:transform>
Example A–6 Remedy_DefaultCategory.xsl

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ns0="http://xmlns.oracle.com/sysman/connector">
    <!--This template creates an incident type ticket with default categorization. The ticket priority is based on event severity. On update, the ticket summary is updated with the latest incident message and severity information. -->
    <xsl:template match="ns0:EMIncident">
        <xsl:choose>
            <xsl:when test="normalize-space(ns0:TicketID) = ''">
                <urn:HelpDesk_Submit_Service xmlns:urn="urn:HPD_IncidentInterface_Create_WS">
                    <!-- EDIT THE TAG VALUES BELOW TO CHANGE HOW A TICKET IS FILLED DURING TICKET CREATION. REFER TO THE REMEDY SERVICE DESK MANUAL FOR DESCRIPTION OF THESE SERVICEDESK SUPPORT DATAFIELDS -->
                    <urn:First_Name><xsl:value-of select="ns0:HDUser"/></urn:First_Name>
                    <urn:Last_Name><xsl:value-of select="ns0:HDUser"/></urn:Last_Name>
                    <urn:Reported_Source>Systems Management</urn:Reported_Source>
                    <urn:Service_Type>Infrastructure Event</urn:Service_Type>
                    <urn:Action>CREATE</urn:Action>
                    <urn:Create_Request/>
                    <urn:Summary>
                        <!-- EDIT THE TAG VALUES BELOW TO CHANGE HOW A TICKET IS FILLED DURING TICKET CREATION. REFER TO THE REMEDY SERVICE DESK MANUAL FOR DESCRIPTION OF THESE SERVICEDESK SUPPORT DATAFIELDS -->
                        <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary,0,"}>
            </xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL')">
                <urn:Impact>1-Extensive/Widespread</urn:Impact>
            </xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
                <urn:Impact>2-Significant/Large</urn:Impact>
            </xsl:when>
            <xsl:otherwise>
                <urn:Impact>3-Moderate/Limited</urn:Impact>
            </xsl:otherwise>
        </xsl:choose>
    </urn:Create_Request/>
</urn:HelpDesk_Submit_Service>
</xsl:template>
</xsl:transform>
```

Ticketing Connector Samples   A-5
Incident updated by Oracle Enterprise Manager Remedy Service Desk Connector

Incident updated by Oracle Enterprise Manager Remedy Service Desk Connector
<urn:Work_Info_Type>Incident Task / Action</urn:Work_Info_Type>
<urn:Work_Info_Date/>
<urn:Work_Info_Source/>
<urn:Work_Info_Locked/>
<urn:Work_Info_View_Access>Public</urn:Work_Info_View_Access>
<urn:Incident_Number>
<xsl:value-of select="ns0:TicketID"/>
</urn:Incident_Number>
<urn:ServiceCI/>
<urn:ServiceCI_ReconID/>
<urn:HPD_CI/>
<urn:HPD_CI_ReconID/>
<urn:HPD_CI_FormName/>
</urn:HelpDesk_Modify_Service>
</xsl:otherwise>
</xsl:template>

<xsl:template name="getDetails">
<xsl:param name="message" />  
<xsl:choose>
<xsl:when test="normalize-space(ns0:SystemAttributes/ns0:UpdatedAttributes) != ''">
<xsl:value-of select="$message" /> for change in attributes :
<xsl:value-of select="ns0:SystemAttributes/ns0:UpdatedAttributes" />
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$message" "/>
</xsl:otherwise>
</xsl:choose>

-----------------------------------------------------------------------------

EM User: <xsl:value-of select="ns0:NotificationRuleOwner"/>

Incident Information:
<xsl:for-each select="ns0:SystemAttributes/ns0:SourceInfo">
Source Name:<xsl:value-of select="./ns0:SourceObjInfo/ns0:ObjName" />
<xsl:choose>
<xsl:when test="normalize-space(./ns0:SourceObjInfo/ns0:ObjOwner) != ''">
Source Owner:<xsl:value-of select="./ns0:SourceObjInfo/ns0:ObjOwner" />
</xsl:when>
</xsl:choose>
Source Type:<xsl:value-of select="./ns0:SourceObjInfo/ns0:SourceObjType"/>
Source SubType:<xsl:value-of select="./ns0:SourceObjInfo/ns0:SourceObjSubType"/>
Target Name: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetName"/>
Target Type: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetType"/>
Target Type Label: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetTypeLabel"/>
Target URL: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetURL"/>
</xsl:for-each>

<!-- LIST ALL THE TARGET PROPERTIES -->
Target Properties:
<xsl:for-each select="ns0:SystemAttributes/ns0:SourceInfo/ns0:TargetInfo/
ns0:TargetProperty"/>
</xsl:for-each>

<!-- EDIT THE FOLLOWING CODE TO LIST A SPECIFIC TARGET PROPERTY, SUCH AS 'Line of Business' -->
<xsl:choose>
<xsl:when test="ns0:SystemAttributes/ns0:SourceInfo/ns0:TargetInfo/ ns0:TargetProperty/ns0:Name='Line of Business'">
  Line of Business: <xsl:value-of select="ns0:SystemAttributes/ ns0:SourceInfo/ns0:TargetInfo/ns0:TargetProperty/ ns0:value"/>
</xsl:when>
</xsl:choose>

Severity: <xsl:value-of select="ns0:SystemAttributes/ns0:Severity"/>
Priority: <xsl:value-of select="ns0:SystemAttributes/ns0:Priority"/>
CreationDate: <xsl:value-of select="ns0:SystemAttributes/ns0:CreationDate"/>
LastUpdatedDate: <xsl:value-of select="ns0:SystemAttributes/ns0:LastUpdatedDate"/>
Owner: <xsl:value-of select="ns0:SystemAttributes/ns0:Owner"/>

<xsl:choose>
  <xsl:when test="normalize-space(ns0:NotificationRuleName) != ''">
    Notification Rule: <xsl:value-of select="ns0:NotificationRuleName"/>
  </xsl:when>
</xsl:choose>

URL: <xsl:value-of select="ns0:SystemAttributes/ns0:IncidentURL"/>
EM Incident Status: <xsl:value-of select="ns0:SystemAttributes/ns0:ResolutionState"/>
EM Acknowledge: <xsl:value-of select="ns0:SystemAttributes/ns0:Acknowledge"/>
EM Auto Close: <xsl:value-of select="ns0:SystemAttributes/ns0:AutoClose"/>
EM Escalation Level: <xsl:value-of select="ns0:SystemAttributes/ns0:EscalationLevel"/>

<xsl:for-each select="ns0:SystemAttributes/ns0:AdditionalDetails">
  <xsl:value-of select="/ns0:VariableName"/>
  : <xsl:value-of select="/ns0:VariableValue"/>
</xsl:for-each>
</xsl:template>
</xsl:transform>

Example A–7 Remedy_DefaultCategory_AutoClose.xsl

<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version='1.0' xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ns0="http://xmlns.oracle.com/sysman/connector">
  <![CDATA[
  <!--
  This template creates an incident type ticket within Remedy Service Desk with
default settings. On update, the worklog is updated with the latest event
message and severity information. The template supports auto closing of
tickets, once the ticket is closed it can not be reopened.-->
  -->

  <xsl:template match="ns0:EMIncident">
    <xsl:choose>
      <xsl:when test="normalize-space(ns0:TicketID) = ''">
        <urn:HelpDesk_Submit_Service xmlns:urn="urn:HPD_IncidentInterface_Create_WS">
          <!-- EDIT THE TAG VALUES BELOW TO CHANGE HOW A TICKET IS FILLED
           DURING TICKET CREATION. REFER TO THE REMEDY SERVICE DESK MANUAL
           FOR DESCRIPTION OF THESE HELPDESK SUPPORT DATAFIELDS--> 
          <!--FIRST_NAME, LAST_NAME VALUES ARE PICKED FROM THE USERNAME VALUE GIVEN
           DURING REMEDY SERVICE DESK CONNECTOR CONFIGURATION. EXAMPLE USERNAME:Demo.-->
        </urn:HelpDesk_Submit_Service>
      </xsl:when>
    </xsl:choose>
  </xsl:template>
</xsl:transform>
<urn:First_Name><xsl:value-of select="ns0:HDUser"/></urn:First_Name>
<urn:choose>
  <urn:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL')">
    <urn:Impact>1-Extensive/Widespread</urn:Impact>
  </urn:when>
  <urn:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
    <urn:Impact>2-Significant/Large</urn:Impact>
  </urn:when>
  <urn:otherwise>
    <urn:Impact>3-Moderate/Limited</urn:Impact>
  </urn:otherwise>
</urn:choose>

<urn:Last_Name><xsl:value-of select="ns0:HDUser"/></urn:Last_Name>
<urn:Reported_Source>Systems Management</urn:Reported_Source>
<urn:Service_Type>Infrastructure Event</urn:Service_Type>
<urn:Action>CREATE</urn:Action>
<urn:Create_Request/>
<urn:Summary>
  <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Summary>
<urn:Notes>
  <xsl:call-template name="getDetails">
    <xsl:with-param name="message">Incident created by Oracle Enterprise Manager Remedy Service Desk Connector</xsl:with-param>
  </xsl:call-template>
</urn:Notes>
<urn:choose>
  <urn:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL') or
  (ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
    <urn:Urgency>1-Critical</urn:Urgency>
  </urn:when>
  <urn:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'WARNING')">
    <urn:Urgency>2-High</urn:Urgency>
  </urn:when>
  <urn:otherwise>
    <urn:Urgency>3-Medium</urn:Urgency>
  </urn:otherwise>
</urn:choose>
<urn:Work_Info_Summary>
  <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Work_Info_Summary>
<urn:Work_Info_Notes>
  <xsl:call-template name="getDetails">
    <xsl:with-param name="message">Incident created by Oracle Enterprise Manager Remedy Service Desk Connector</xsl:with-param>
  </xsl:call-template>
</urn:Work_Info_Notes>
<urn:Work_Info_Type>Incident Task / Action</urn:Work_Info_Type>
<urn:Work_Info_Date/>
</urn:HelpDesk_Submit_Service>
<urn:HelpDesk_Modify_Service xmlns="urn:HPD_IncidentInterface_Custom_WS">
  <urn:Categorization_Tier_1/>
  <urn:Categorization_Tier_2/>
</urn:HelpDesk_Modify_Service>
<urn:Categorization_Tier_3/></urn:Categorization_Tier_3>
<urn:Closure_Manufacturer></urn:Closure_Manufacturer>
<urn:Closure_Product_Category_Tier1></urn:Closure_Product_Category_Tier1>
<urn:Closure_Product_Category_Tier2></urn:Closure_Product_Category_Tier2>
<urn:Closure_Product_Category_Tier3></urn:Closure_Product_Category_Tier3>
<urn:Closure_Product_Model_Version></urn:Closure_Product_Model_Version>
<urn:Closure_Product_Name></urn:Closure_Product_Name>
<!--EDIT THE Company TAG BELOW TO ADD A Company NAME THAT IS ASSOCIATED
WITH FIRST_NAME,LAST_NAME TAGS ON THE REMEDY -->
<urn:Company>My Company</urn:Company>
<urn:Summary>
    <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Summary>
<urn:Notes>
    <xsl:choose>
        <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
            <urn:Resolution>
                Incident closed by Oracle Enterprise Manager Remedy Service Desk Connector due to change in severity of the associated alert. Severity: <xsl:value-of select="ns0:SystemAttributes/ns0:Severity"/>
                Message: <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
            </urn:Resolution>
        </xsl:when>
        <xsl:otherwise>
            <urn:Resolution>
                Incident updated by Oracle Enterprise Manager Remedy Service Desk Connector <xsl:with-param name="message">Incident updated by Oracle Enterprise Manager Remedy Service Desk Connector</xsl:with-param>
            </urn:Resolution>
        </xsl:otherwise>
    </xsl:choose>
</urn:Notes>
<urn:Impact>
    <xsl:choose>
        <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL')">
            1-Extensive/Widespread
        </xsl:when>
        <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
            2-Significant/Large
        </xsl:when>
        <xsl:otherwise>
            3-Moderate/Limited
        </xsl:otherwise>
    </xsl:choose>
</urn:Impact>
<urn:Manufacturer></urn:Manufacturer>
<urn:Product_Categorization_Tier_1></urn:Product_Categorization_Tier_1>
<urn:Product_Categorization_Tier_2></urn:Product_Categorization_Tier_2>
<urn:Product_Categorization_Tier_3></urn:Product_Categorization_Tier_3>
<urn:Product_Model_Version></urn:Product_Model_Version>
<urn:Product_Name></urn:Product_Name>
<urn:Reported_Source>Systems Management</urn:Reported_Source>
0, 100)"/>
    </urn:Resolution>
  </urn:when>
  <urn:Resolution/>
</urn:when>
</urn:Resolution>
</urn:otherwise>
</urn:Resolution>
</urn:choose>
<urn:Resolution_Category/>
<urn:Resolution_Category_Tier_2/>
<urn:Resolution_Category_Tier_3/>
<urn:Resolution_Method/>
<urn:Service_Type>Infrastructure Event</urn:Service_Type>

<xsl:choose>
  <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
    <urn:Status>Closed</urn:Status>
  </xsl:when>
  <urn:Status/>
</urn:when>
</urn:otherwise>
</urn:otherwise>
</urn:choose>

<xsl:choose>
  <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL') or
  (ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">    
    <urn:Urgency>1-Critical</urn:Urgency>
  </xsl:when>
  <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'WARNING')">
    <urn:Urgency>2-High</urn:Urgency>
  </xsl:when>
  <urn:Urgency/>    
</urn:otherwise>
</urn:otherwise>
</urn:choose>
<xsl:choose>
  <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
    <urn:Action>MODIFY</urn:Action>
  </xsl:when>
  <urn:Action/>
</urn:when>
</urn:otherwise>
</urn:otherwise>
</urn:choose>
<urn:Work_Info_Summary/>
<xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0,
100)"/>
</urn:Work_Info_Summary>
<xsl:choose>
  <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
    <urn:Work_Info_Notes>
      <xsl:call-template name="getDetails">
        <xsl:with-param name="message">Incident closed by Oracle
        Enterprise Manager Remedy Service Desk Connector due to change in
        associated Incident</xsl:with-param>
      </xsl:call-template>
    </urn:Work_Info_Notes>
  </xsl:when>
  <urn:Work_Info_Notes/>
</urn:when>
</urn:otherwise>
</urn:otherwise>
</urn:choose>
<xsl:choose>
  <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
    <urn:Work_Info_Notes>
      <xsl:call-template name="getDetails">
        <xsl:with-param name="message">Incident updated due to change in
        associated Incident</xsl:with-param>
      </xsl:call-template>
    </urn:Work_Info_Notes>
  </xsl:when>
  <urn:Work_Info_Notes/>
</urn:when>
</urn:otherwise>
</urn:otherwise>
</urn:choose>
<urn:Work_Info_Type>Incident Task / Action</urn:Work_Info_Type>
<urn:Work_Info_Date/>
<urn:Work_Info_Source/>
<urn:Work_Info_Locked></urn:Work_Info_Locked>
<urn:Work_Info_View_Access>Public</urn:Work_Info_View_Access>
<urn:Incident_Number>
<xsl:value-of select="ns0:TicketID"/>
</urn:Incident_Number>
<urn:ServiceCI></urn:ServiceCI>
<urn:ServiceCI_ReconID></urn:ServiceCI_ReconID>
<urn:HPD_CI></urn:HPD_CI>
<urn:HPD_CI_ReconID></urn:HPD_CI_ReconID>
<urn:HPD_CI_FormName></urn:HPD_CI_FormName>
</urn:HelpDesk_Modify_Service>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<xsl:template name="getDetails">
<xsl:param name="message" />
<xsl:choose>
<xsl:when test="normalize-space(ns0:SystemAttributes/ns0:UpdatedAttributes) != ''">
<xsl:value-of select="$message" /> for change in attributes :
<xsl:value-of select="ns0:SystemAttributes/ns0:UpdatedAttributes" />
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$message" />
</xsl:otherwise>
</xsl:choose>

EM User: <xsl:value-of select="ns0:NotificationRuleOwner"/>
Incident Information:
<xsl:for-each select="ns0:SystemAttributes/ns0:SourceInfo">
Source Name:<xsl:value-of select="./ns0:SourceObjInfo/ns0:ObjName" />
<xsl:choose>
<xsl:when test="normalize-space(./ns0:SourceObjInfo/ns0:ObjOwner) != ''">
Source Owner:<xsl:value-of select="./ns0:SourceObjInfo/ns0:ObjOwner" />
</xsl:when>
</xsl:choose>
Source Type:<xsl:value-of select="./ns0:SourceObjInfo/ns0:SourceObjType"/>
Source SubType:<xsl:value-of select="./ns0:SourceObjInfo/ns0:SourceObjSubType"/>
Target Name: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetName"/>
Target Type: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetType"/>
Target Type Label: <xsl:value-of select="./ns0:TargetInfo/ns0:TargetTypeLabel"/>
Target URL:<xsl:value-of select="./ns0:TargetInfo/ns0:TargetURL"/>
</xsl:for-each>
<!-- LIST ALL THE TARGET PROPERTIES -->
Target Properties:
<xsl:for-each select="ns0:SystemAttributes/ns0:SourceInfo/ns0:TargetInfo/ns0:TargetProperty" >
<xsl:text>&#xa;    </xsl:text>
</xsl:for-each>
<!-- EDIT THE FOLLOWING CODE TO LIST A SPECIFIC TARGET PROPERTY, SUCH AS 'Line of Business' -->
<xsl:choose>
<xsl:when test="ns0:SystemAttributes/ns0:SourceInfo/ns0:TargetInfo/ns0:TargetProperty/ns0:Name='Line of Business'">
Line of Business: <xsl:value-of select="ns0:ns0:SystemAttributes/
Example A–8 Remedy_DefaultCategory_AutoResolve.xsl

<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmns:ns0="http://xmlns.oracle.com/sysman/connector">
  <!-- This template creates an incident type ticket within Remedy Service Desk with default settings. On update, the worklog is updated with the latest incident message and severity information. The ticket is set to status Resolved if the associated alert has cleared. Ticket can be reopened if a severity occurred within the grace period. If the ticket is not reopened for 15 days, ticket will be closed by incident management. -->
  <xsl:template match="ns0:EMIncident">
    <xsl:choose>
      <xsl:when test="normalize-space(ns0:TicketID) = ''">
        <urn:HelpDesk_Submit_Service xmlns:urn="urn:HPD_IncidentInterface_Create_WS">
          <!-- EDIT THE TAG VALUES BELOW TO CHANGE HOW A TICKET IS FILLED DURING TICKET CREATION. REFER TO THE REMEDY SERVICE DESK MANUAL FOR DESCRIPTION OF THESE HELPDESK SUPPORT DATAFIELDS-->
          <!--FIRST_NAME, LAST_NAME VALUES ARE PICKED FROM THE USERNAME VALUE GIVEN DURING REMEDY SERVICE DESK CONNECTOR CONFIGURATION. EXAMPLE USERNAME:Demo.-->
          <urn:First_Name><xsl:value-of select="ns0:HDUser"/></urn:First_Name>
        </urn:HelpDesk_Submit_Service>
      </xsl:when>
    </xsl:choose>
  </xsl:template>
</xsl:transform>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
  <urn:Impact>2-Significant/Large</urn:Impact>
</xsl:when>
<xsl:otherwise>
  <urn:Impact>3-Moderate/Limited</urn:Impact>
</xsl:otherwise>
</xsl:choose>
<urn:Last_Name><xsl:value-of select="ns0:HDUser"/></urn:Last_Name>
<urn:Reported_Source>Systems Management</urn:Reported_Source>
<urn:Service_Type>Infrastructure Event</urn:Service_Type>
<urn:Action>CREATE</urn:Action>
<urn:Create_Request/>
<urn:Summary>
  <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Summary>
<urn:Notes>
  <xsl:call-template name="getDetails">
    <xsl:with-param name="message">Incident created by Oracle Enterprise
Manager Remedy Service Desk Connector</xsl:with-param>
  </xsl:call-template>
</urn:Notes>
<urn:Urgency>
  <xsl:choose>
    <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL') or
                   (ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
      1-Critical
    </xsl:when>
    <xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'WARNING')">
      2-High
    </xsl:when>
    <xsl:otherwise>
      3-Medium
    </xsl:otherwise>
  </xsl:choose>
</urn:Urgency>
<urn:Work_Info_Summary>
  <xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Work_Info_Summary>
<urn:Work_Info_Notes>
  <xsl:call-template name="getDetails">
    <xsl:with-param name="message">Incident created by Oracle Enterprise
Manager Remedy Service Desk Connector</xsl:with-param>
  </xsl:call-template>
</urn:Work_Info_Notes>
<urn:Work_Info_Type>Incident Task / Action</urn:Work_Info_Type>
<urn:Work_Info_Date/>
</urn:HelpDesk_Submit_Service>
<xsl:otherwise>
  <urn:HelpDesk_Modify_Status_Service xmlns:urn="urn:HPD_IncidentInterface_Custom_WS">
    <urn:Categorization_Tier_1/>
    <urn:Categorization_Tier_2/>
    <urn:Categorization_Tier_3/>
    <urn:Closure_Manufacturer/>
    <urn:Closure_Product_Category_Tier1/>*
  </urn:HelpDesk_Modify_Status_Service>
</xsl:otherwise>
<!--EDIT THE Company TAG BELOW TO ADD A Company NAME THAT IS ASSOCIATED WITH FIRST_NAME,LAST_NAME TAGS ON THE REMEDY -->
<urn:Company>My Company</urn:Company>

<xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Summary>

<xsl:choose>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
<urn:Resolution>Incident resolved by Oracle Enterprise Manager Remedy Service Desk Connector</urn:Resolution>
</xsl:when>
<xsl:when test="ns0:ReopenTicket = 'Yes'">
<urn:Resolution>Incident reopened by Oracle Enterprise Manager Remedy Service Desk Connector</urn:Resolution>
</xsl:when>
<xsl:otherwise>
<urn:Resolution>Incident updated by Oracle Enterprise Manager Remedy Service Desk Connector</urn:Resolution>
</xsl:otherwise>
</xsl:choose>

<xsl:choose>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL')">
<urn:Impact>1-Extensive/Widespread</urn:Impact>
</xsl:when>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
<urn:Impact>2-Significant/Large</urn:Impact>
</xsl:when>
<xsl:otherwise>
<urn:Impact>3-Moderate/Limited</urn:Impact>
</xsl:otherwise>
</xsl:choose>

Incident resolved by Oracle Enterprise Manager Remedy Service Desk Connector due to change in severity of the associated alert. Severity:
<xsl:value-of select="ns0:SystemAttributes/ns0:Severity"/>
Message:
<xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Resolution>
</xsl:when>
<xsl:otherwise>
<urn:Resolution></urn:Resolution>
</xsl:otherwise>
</xsl:choose>
<urn:Resolution_Category/>
<urn:Resolution_Category_Tier_2/>
<urn:Resolution_Category_Tier_3/>
<urn:Resolution_Method/>
<urn:Service_Type>Infrastructure Event</urn:Service_Type>
<xsl:choose>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
<urn:Status>Resolved</urn:Status>
</xsl:when>
<xsl:otherwise>
<urn:Status></urn:Status>
</xsl:otherwise>
</xsl:choose>
<xsl:choose>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'FATAL') or
(nso:SystemAttributes/ns0:SeverityCode = 'CRITICAL')">
<urn:Urgency>1-Critical</urn:Urgency>
</xsl:when>
<xsl:when test="ns0:SystemAttributes/ns0:SeverityCode = 'WARNING'">
<urn:Urgency>2-High</urn:Urgency>
</xsl:when>
<xsl:otherwise>
<urn:Urgency>3-Medium</urn:Urgency>
</xsl:otherwise>
</xsl:choose>
<urn:Action>MODIFY</urn:Action>
<urn:Work_Info_Summary>
<xsl:value-of select="substring(ns0:SystemAttributes/ns0:Summary, 0, 100)"/>
</urn:Work_Info_Summary>
<xsl:choose>
<xsl:when test="(ns0:SystemAttributes/ns0:SeverityCode = 'CLEAR')">
<urn:Work_Info_Notes>
<xsl:call-template name="getDetails">
<xsl:with-param name="message">Incident resolved by Oracle Enterprise Manager Remedy Service Desk Connector due to change in associated Incident</xsl:with-param>
</xsl:call-template>
</urn:Work_Info_Notes>
</xsl:when>
<xsl:when test="ns0:ReopenTicket = 'Yes'">
<urn:Work_Info_Notes>
<xsl:call-template name="getDetails">
<xsl:with-param name="message">Incident reopened because the associated event re-triggered within the grace period</xsl:with-param>
</xsl:call-template>
</urn:Work_Info_Notes>
</xsl:when>
</xsl:choose>
Incident updated by Oracle Enterprise Manager Remedy Service Desk Connector

EM User: [EM User Name]

Source Name: [Source Name]
Source Type: [Source Type]
Source Owner: [Source Owner]
Source SubType: [Source SubType]
Example A–9  connectorDeploy.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
<xsd:element name="ManagementConnector">
  <xsd:annotation>
    <xsd:documentation>Deployment Descriptor for Management Connectors</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Name" type="StringT64">
        <xsd:annotation>
          <xsd:documentation>The name of the connector type.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="Version" type="VersionT">
        <xsd:annotation>
          <xsd:documentation>Version of the connector type.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="EMCompatibleVersion" type="VersionT">
        <xsd:annotation>
          <xsd:documentation>The EM compatibility version of the connector type.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="Description" type="StringT256">
        <xsd:annotation>
          <xsd:documentation>The description of the connector type.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="Category">
        <xsd:annotation>
          <xsd:documentation>The category of the connector type. It must be one of the three values listed next.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="EventConnector"/>
            <xsd:enumeration value="TicketingConnector"/>
            <xsd:enumeration value="ChangeManagementConnector"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <!-- NewTargetType is for EventConnector only. -->
      <xsd:element name="NewTargetType" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>New target type definition for event connectors. This target type will be registered with Enterprise Manager and target instances can
be created subsequently, including a default target. These targets are used to accommodate external events.
</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element name="TargetTypeName" type="StringStrictT64">
<xsd:annotation>
<xsd:documentation>
The name of the target type.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="TargetTypeDisplayName" type="StringT128">
<xsd:annotation>
<xsd:documentation>
The name of the target type, as shown on UI.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="DefaultTargetName" type="StringStrictT256">
<xsd:annotation>
<xsd:documentation>
The name of the default target of the target type. The default target will be used as a generic bucket to hold external events.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="DefaultTargetDisplayName" type="StringT256">
<xsd:annotation>
<xsd:documentation>
The name of the default target of the target type, to be displayed on UI.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="SOAPHeaderAuthentication" type="SOAPHeaderAuthenticationType" minOccurs="0">
<xsd:annotation>
<xsd:documentation>
Specification for SOAP Header authentication.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="HTTPBasicAuthentication" type="UsernamePasswordAuthenticationType" minOccurs="0">
<xsd:annotation>
<xsd:documentation>
Specification for HTTP basic authentication.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="UserNameTokenAuthentication" type="UsernamePasswordAuthenticationType" minOccurs="0">
<xsd:annotation>
<xsd:documentation>
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:annotation>
  <xsd:documentation>
    Specification for Username Token authentication.
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="ConfigVariable" type="ConfigVariableType"
  minOccurs="0" maxOccurs="20">
  <xsd:annotation>
    The variables used during connector configuration. These variables
    are required by external system to complete connector
    configuration, which includes registering with the external system. For instance,
    one configuration variable can be the resolution state required by
    Microsoft Operation Manager.
  </xsd:annotation>
</xsd:element>

<xsd:element name="ConnectivityTestVariable" type="ConfigVariableType"
  minOccurs="0">
  <xsd:annotation>
    An optional variable used to test connection to an external
    server.
  </xsd:annotation>
</xsd:element>

<xsd:element name="Service" type="ServiceType" maxOccurs="20">
  <xsd:annotation>
    Specification for web services, which define how connector
    framework
    can communicate with external system.
  </xsd:annotation>
</xsd:element>

<xsd:element name="ExternalURL" type="ExternalURLType" minOccurs="0">
  <xsd:annotation>
    Specification for the URL link to the external server, including
    the URL pattern and server specific variables. It is used to
    provide links
    to external server for viewing ticket details.
  </xsd:annotation>
</xsd:element>

<xsd:element name="TemplateRegistration" type="TemplateRegistrationType"
  minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    Specification for template registration. A template is registered
    based on the information provided in the element. A connector
    deployment
description can have an optional list of up to 50 template
    registration
    elements.
  </xsd:annotation>
</xsd:element>
<xsd:complexType name="ServiceType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for a web service.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="Method">
      <xsd:annotation>
        <xsd:documentation>
          The name of the web service method. Each connector category has a predefined set of methods as defined next.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <!-- event connector: -->
          <xsd:enumeration value="setup"/>
          <xsd:enumeration value="initialize"/>
          <xsd:enumeration value="getNewAlerts"/>
          <xsd:enumeration value="getUpdatedAlerts"/>
          <xsd:enumeration value="acknowledgeAlerts"/>
          <xsd:enumeration value="updateAlerts"/>
          <xsd:enumeration value="createEvent"/>
          <xsd:enumeration value="updateEvent"/>
          <xsd:enumeration value="uninitialize"/>
          <xsd:enumeration value="cleanup"/>
          <!-- ticketing connector: -->
          <xsd:enumeration value="createTicket"/>
          <xsd:enumeration value="updateTicket"/>
          <xsd:enumeration value="getTicket"/>
          <!-- change management connector: -->
          <xsd:enumeration value="publishCS"/>
          <xsd:enumeration value="updateChangeRequest"/>
          <xsd:enumeration value="getChangeRequest"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="WebServiceEndpoint" type="StringT256">
      <xsd:annotation>
        <xsd:documentation>
          The web service end point indicating a specific location for accessing a service.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SOAPAction" type="StringT64" minOccurs="0">
      <xsd:annotation>
        <xsd:documentation>
          The SOAP action which carries out the web service call for the method.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
The type of SOAP over HTTP binding. Choose from one of the four options defined next.

This section defines a complex type for SOAP Header Authentication.

The username of the authentication.

The password of the authentication.

An optional list of extra authentication variables besides username and password.

A SOAP header string serving as template for the SOAP header. It is to be updated with user inputs for variables defined above and bound with a HTTP request.
This section defines a complex type for Username Password authentication.

```xml
<complexType name="UsernamePasswordAuthentication">
  <sequence>
    <element name="Username" type="ConfigVariableType">
      <documentation>
        The username of the authentication.
      </documentation>
    </element>
    <element name="Password" type="ConfigVariableType">
      <documentation>
        The password of the authentication.
      </documentation>
    </element>
  </sequence>
</complexType>
```

```xml
<complexType name="ConfigVariableType">
  <sequence>
    <element name="VariableName" type="StringStrictT32">
      <documentation>
        Name of the variable.
      </documentation>
    </element>
    <element name="DisplayName" type="StringT64">
      <documentation>
        Name of the variable used for display on UI.
      </documentation>
    </element>
  </sequence>
  <attribute name="required" type="xsd:boolean" default="false">
    <documentation>
      A Flag indicating whether or not the variable is mandatory.
    </documentation>
  </attribute>
</complexType>
```

```xml
<complexType name="ExternalURLType">
  <sequence>
    <element name="Pattern" type="StringT256">
      <documentation>
        This section defines a complex type for external URL.
      </documentation>
    </element>
  </sequence>
</complexType>
```
<xsd:documentation>
The URL pattern used to format links to the external server.
</xsd:documentation>
</xsd:annotation>
<xsd:element>
  <xsd:annotation>
    <xsd:documentation>
      An optional list of configuration variableeles representing the details of the external server. They are used for constructing links to the server based on the URL pattern.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemplateRegistrationType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for template registration metadata which is used to register templates during connector deployment.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="FileName" type="StringT256">
      <xsd:annotation>
        <xsd:documentation>
          The template file name.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="InternalName" type="StringStrictT128">
      <xsd:annotation>
        <xsd:documentation>
          A name representing the template in the connector framework.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="TemplateName" type="StringStrictT128">
      <xsd:annotation>
        <xsd:documentation>
          The template display name to be used on UI.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="TemplateType">
      <xsd:annotation>
        <xsd:documentation>
          The template type as one of the three options defined next.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">  
          <xsd:enumeration value="InboundXSL"/>  
          <xsd:enumeration value="OutboundXSL"/>  
          <xsd:enumeration value="OutboundXML"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="EMEvent">
  <xsd:annotation>
    <xsd:documentation>
      This section defines an EM event made available through the connector framework.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ConnectorGUID" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            A unique ID to identify the connector used to forward the EM event to the targeted external event system.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="ExternalEventID" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>
            The ID to identify the event created in the external event system.
            It is generated in the external system and used to update the external event.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="NotificationRuleOwner" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            The owner of the notification rule which delivers the event.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="NotificationRuleName" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            The name of the notification rule which delivers the event.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

Example A–10  EMEvent.xsd

<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://xmlns.oracle.com/sysman/connector"
  targetNamespace="http://xmlns.oracle.com/sysman/connector"
  elementFormDefault="qualified">
  <xsd:include schemaLocation="connectorCommon.xsd"/>
  <xsd:element name="EMEvent">
    <xsd:annotation>
      <xsd:documentation>
        This section defines an EM event made available through the connector framework.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ConnectorGUID" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              A unique ID to identify the connector used to forward the EM event to the targeted external event system.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="ExternalEventID" type="xsd:string" minOccurs="0">
          <xsd:annotation>
            <xsd:documentation>
              The ID to identify the event created in the external event system.
              It is generated in the external system and used to update the external event.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="NotificationRuleOwner" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              The owner of the notification rule which delivers the event.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="NotificationRuleName" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              The name of the notification rule which delivers the event.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
defined in connectorDeploy.xsd.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="Property" type="PropertyType" minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    <xsd:documentation>
      An optional list of up to 50 property variables as defined in
      connectorCommon.xsd.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SystemAttributes" type="EventSystemAttributesType">
  <xsd:annotation>
    <xsd:documentation>
      A list of attributes for events as defined by EM event system.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="EventClassSpecificAttributes">
  <xsd:annotation>
    <xsd:documentation>
      A list of attributes for events that are specific to the event
      class.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:complexType>
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="200">
      <xsd:element name="StringAttribute" type="StringValueType">
        <xsd:annotation>
          <xsd:documentation>
            A String attribute.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="NumberAttribute" type="StringValueType">
        <xsd:annotation>
          <xsd:documentation>
            A Number attribute.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="RawAttribute" type="StringValueType">
        <xsd:annotation>
          <xsd:documentation>
            An attribute of type Raw.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="EventClass" type="xsd:string">
  <xsd:documentation>
    The class of the event. For instance, target availability event, metric alert event, job status change event.
  </xsd:documentation>
</xsd:element>
The ID to identify a single event instance.

ID to identify a sequence of events which share the same event life.

Date when the event occurred.

The date timestamp when the EM event publishing system is reporting the event.

The display timezone region associated with the event. Event publishers can specify the time zone the event should be displayed in.

Name of the event.

Severity level of the current event. The value changes based on local language setting.
Internal Severity value of the current event.
</xsd:documentation>
</xsd:element>
<xsd:element name="SourceInfo" type="SourceInfoType">
  <xsd:annotation>
    <xsd:documentation>
The source information of the EM subsystems or components that raises the event.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Message" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
    A description of the event.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="ActionMessage" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
The action message for the event that helps diagnosing the issue.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="EventURL" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
    A URL of the event on EM incident console.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="AutoClose" type="xsd:boolean">
  <xsd:annotation>
    <xsd:documentation>
    A flag indicating if an event is auto closed by the system, or it has to be manually closed by users.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="EventCategory" type="xsd:string" minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    <xsd:documentation>
    An optional list of event categories to which the event belongs.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="StringValueType">
  <xsd:annotation>
    <xsd:documentation>
    This section defines a complex type for a general name/value pair, both in String.
    </xsd:documentation>
  </xsd:annotation>
</xsd:complexType>
Example A–11  EMIncident.xsd

<xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="connectorCommon.xsd"/>
    <xsd:include schemaLocation="EMEvent.xsd"/>
    <xsd:element name="EMIncident">
        <xsd:annotation>
            <xsd:documentation>
                This section provides a data structure based on EM incidents for all
ticketing actions.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            ...
<xsd:sequence>
  <xsd:element name="ConnectorGUID" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        A unique ID to identify the connector that is processing
        the incident.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="TicketID" type="xsd:string" minOccurs="0">
    <xsd:annotation>
      <xsd:documentation>
        The ID to identify the ticket created in the external
ticketing system.
        It is generated in the external system and used to update
        the ticket.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="HDUser" type="xsd:string" minOccurs="0">
    <xsd:annotation>
      <xsd:documentation>
        HelpDesk user name provided from UI during connector
        configuration.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="NotificationRuleOwner" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        The owner of the notification rule which generated the
        incident.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="NotificationRuleName" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        The name of the notification rule which generated the
        incident.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="ReopenTicket" type="xsd:string" minOccurs="0">
    <xsd:annotation>
      <xsd:documentation>
        The identifier of the previous ticket that should be
        re-opened.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="ConnectorVariable" type="VariableType"
    minOccurs="0" maxOccurs="50">
    <xsd:annotation>
      <xsd:documentation>
        An optional list of up to 50 connector variables that contain
        name/value pairs.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:sequence>
<xsd:element name="Property" type="PropertyType" minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    <xsd:documentation>
      An optional list of up to 50 property variables as defined in connectorCommon.xsd.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="SystemAttributes" type="IncidentSystemAttributesType">
  <xsd:annotation>
    <xsd:documentation>
      A list of attributes for incidents as defined by EM event system.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="HasEMEvent" type="xsd:boolean" minOccurs="1" maxOccurs="1">
  <xsd:annotation>
    <xsd:documentation>
      Flag to check an EM Event.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="EMEvent" type="EMEventType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      A list of attributes associated with EM Event.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

</xsd:complexType>
</xsd:element>

<xsd:complexType name="IncidentSystemAttributesType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for incident attributes provided by EM event system.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="IncidentID" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          The ID of an incident.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceInfo" type="SourceInfoType" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>
          The source information of the EM subsystems or components that raises the incident.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="IncidentURL" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      A URL to the incident on EM.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="AutoClose" type="xsd:boolean">
  <xsd:annotation>
    <xsd:documentation>
      A flag indicating if an incident is auto closed by the system, or it has to be manually closed by users.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="TicketStatus" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      If an external ticket is associated with the incident, the status of the ticket as assigned at an external help desk system, and updated in EM.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="Owner" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      User to whom the incident is assigned to resolve the issue.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="ResolutionState" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      The attribute used to track where the incident is in terms of resolution. For instance, it can be "new" or "closed".
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="Acknowledge" type="xsd:boolean">
  <xsd:annotation>
    <xsd:documentation>
      A flag indicating whether or not the incident has been acknowledged. Acknowledgement is simply a way for an administrator to indicate that they have viewed the incident and take ownership of it.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="Escalated" type="xsd:boolean">
  <xsd:annotation>
    <xsd:documentation>
      A flag indicating whether or not the incident has been escalated.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="EscalationLevel" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      The hierarchical level of escalation that has been made to this incident.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="Priority" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      The priority order in which the issue should be resolved.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="Summary" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      A text summary of the incident.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="CreationDate" type="xsd:dateTime">
  <xsd:annotation>
    <xsd:documentation>
      The time when the incident is created by associating event to incident.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="LastUpdatedDate" type="xsd:dateTime">
  <xsd:annotation>
    <xsd:documentation>
      The time when the incident is last updated. The incident update includes changes to any of the tracking attributes or changes to the associated events and event sequence.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="Category" type="xsd:string" minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    <xsd:documentation>
      An optional list of categories of the incidents.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="AdditionalDetails" type="VariableType" minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    <xsd:documentation>
      Additional Incident details.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="LastModifiedBy" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      User who last modified the incident.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
Example A–12  externalEvent.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
    <xsd:element name="ExternalEvent">
        <xsd:annotation>
            <xsd:documentation>
                This section defines the attribute requirement of an external event for the connector framework to process it.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="SystemAttributes" type="ExternalEventSystemAttributesType">
                    <xsd:annotation>
                        <xsd:documentation>
                            Attributes to capture general information about the external event system. These attributes are system-specific, with all events from the same external system sharing the same system attributes.
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="EventClassAttributes" type="ExternalEventClassAttributesType">
                    <xsd:annotation>
                        <xsd:documentation>
                            Attributes to capture specific information required for the event as defined in the event class.
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```
This section defines a complex type for system attributes required for all external events.

<xsd:complexType name="ExternalEventSystemAttributesType">
  <xsd:annotation>
    <xsd:documentation>
      Name of the event.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="eventName" type="xsd:string" minOccurs="0">
      <xsd:annotation>
        Name of the event.
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="category" type="xsd:string" minOccurs="0" maxOccurs="50">
      <xsd:annotation>
        The event category to which the event belongs.
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="targetName" type="xsd:string">
      <xsd:annotation>
        Name of the target on which event was generated. It refers to an entity in external systems similar to an EM target.
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="targetType" type="xsd:string">
      <xsd:annotation>
        The type of the target. Target types defined for event connectors are used. See connectorDeploy.xsd.
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="occurrenceDate" type="xsd:dateTime">
      <xsd:annotation>
        Date when the event occurred.
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="detectedDate" type="xsd:dateTime">
      <xsd:annotation>
        Date when the event was last detected.
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="autoClose" type="xsd:boolean">
      <xsd:annotation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
A flag indicating if an event is auto closed by the system, or it has to be manually closed by users.

A description of the event.

Severity level of the event.

This section defines a complex type for class specific attributes required for all external events in the class.

ID used in external system to identify the event.

Optional rule ID that delivered the event in the external system.

Optional host information from external system where event was generated.

Optional source information from the external system.
<xsd:element name="external_severity" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      Severity level of the event on external system.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="external_status" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      Status of the event on external system.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field1" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      An optional field.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field2" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      An optional field.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field3" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      An optional field.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field4" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      An optional field.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field5" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      An optional field.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
</xsd:complexType>
</xsd:schema>

Example A–13  connectorCommon.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://xmlns.oracle.com/sysman/connector"
  targetNamespace="http://xmlns.oracle.com/sysman/connector"
  elementFormDefault="qualified">
<xsd:include schemaLocation="externalEvent.xsd"/>
<xsd:complexType name="SourceInfoType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for Source Information.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="SourceObjInfo" type="SourceObjInfoType" minOccurs="0">
      <xsd:annotation>
        <xsd:documentation>
          This element defines the data structure for the source object, the
          EM subsystem or component, that raises an EM event or an incident.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="TargetInfo" type="TargetInfoType" minOccurs="0">
      <xsd:annotation>
        <xsd:documentation>
          The element defines the data structure for an EM target as related
to the connector framework.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SourceObjInfoType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for Source Object Information.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="ObjID" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          The unique ID to identify the source object.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="ObjName" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          The name of the source object.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="ObjOwner" type="xsd:string" minOccurs="0">
      <xsd:annotation>
        <xsd:documentation>
          The owner of the source object.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceObjType" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          The type of the source object.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SourceInfoType">
  <xsd:annotation>
    <xsd:documentation>
The section defines a complex type for source information.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="SourceGUID" type="xsd:string">  
      <xsd:annotation>
        <xsd:documentation>
          A unique GUID for the source.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceName" type="xsd:string"> 
      <xsd:annotation>
        <xsd:documentation>
          Name of the source.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceType" type="xsd:string"> 
      <xsd:annotation>
        <xsd:documentation>
          Type of the source.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceTypeLabel" type="xsd:string"> 
      <xsd:annotation>
        <xsd:documentation>
          The display label of the source type.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceURL" type="xsd:string"> 
      <xsd:annotation>
        <xsd:documentation>
          The URL of the source.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="SourceProperty" type="PropertyType" minOccurs="0" maxOccurs="50"> 
      <xsd:annotation>
        <xsd:documentation>
          An optional list of properties for the source.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PropertyType">
  <xsd:documentation>
  This section defines a complex type for a property attribute.
  </xsd:documentation>
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string">
      <xsd:documentation>
      A string name defining a property attribute.
      </xsd:documentation>
    </xsd:element>
    <xsd:element name="Value" type="xsd:string" nillable="true">
      <xsd:documentation>
      A non-null string value.
      </xsd:documentation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="VariableType">
  <xsd:documentation>
  This section defines a complex type for a general variable.
  </xsd:documentation>
  <xsd:sequence>
    <xsd:element name="VariableName" type="StringStrictT32">
      <xsd:documentation>
      Name of the variable. It has to be a string containing 1 or upto 32 upper case or lower case letters or numbers.
      </xsd:documentation>
    </xsd:element>
    <xsd:element name="VariableValue" type="StringT2048">
      <xsd:documentation>
      Value of the variable. It has to be a string containing 1 or upto 2048 characters.
      </xsd:documentation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="GetAlertsResponse">
  <xsd:documentation>
  This section defines a complex type for responses to a getAlerts request.
  </xsd:documentation>
  <xsd:sequence>
    <xsd:element name="Alert" minOccurs="0" maxOccurs="200">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:annotation>
  <xsd:documentation>
  The individual alerts contained in the response. A response may have
  upto 200 alerts.
  </xsd:documentation>
</xsd:annotation>
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="ExternalEvent">
      <xsd:annotation>
        <xsd:documentation>
        Details of the external event in the alert, as defined in ExternalEvent.xsd.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="InstanceVariable" type="VariableType" minOccurs="0" maxOccurs="50">
      <xsd:annotation>
        <xsd:documentation>
        A list of instance variables for the alert.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ConnectorVariablesType">
  <xsd:annotation>
    <xsd:documentation>
    This section defines a complex type for connector variables. An element of type ConnectorVariablesType may have up to 50 connector variables, as defined next.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="ConnectorVariable" type="VariableType" minOccurs="0" maxOccurs="50">
      <xsd:annotation>
        <xsd:documentation>
        A connector variable as a name/value pair.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="StringT64">
  <xsd:annotation>
    <xsd:documentation>
    This section defines a simple type for a String with maximum length of 64 bytes.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="64"/>
  </xsd:restriction>
<xsd:simpleType name="StringT128">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 128 bytes.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="128"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringT256">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 256 bytes.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="256"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringT512">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 512 bytes.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="512"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringT2048">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 2048 bytes.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="2048"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringStrictT16">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 16 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
This section defines a simple type for a String with maximum length of 32 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.

This section defines a simple type for a String with maximum length of 64 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.

This section defines a simple type for a String with maximum length of 128 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.

This section defines a simple type for a String with maximum length of 256 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.
Example A–14  createTicket_response.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">

<xsd:include schemaLocation="connectorCommon.xsd"/>
<xsd:element name="CreateTicketResponse">
    <xsd:annotation>
        <xsd:documentation>
            The response from an external system upon a createTicket request. It must contain a ticket ID, along with an optional list of instance variables.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="ticketID" type="StringT128">
                <xsd:annotation>
                    <xsd:documentation>
                        The ID to identify the ticket created in the external ticketing system. It is generated in the external system upon a create ticket request and is used to update the ticket in the future.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="InstanceVariable" type="VariableType" minOccurs="0" maxOccurs="50">
                <xsd:annotation>
                    <xsd:documentation>
                        An optional list of name/value pairs returned by the external system.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
Example A–15  getTicket_request.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="connectorCommon.xsd"/>
    <xsd:element name="getTicketRequest">
        <xsd:annotation>
            <xsd:documentation>
                The request to an external system to test connection. It must contain an existing ticket ID from the targeted external system.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ticketID" type="StringT128">
                    <xsd:annotation>
                        <xsd:documentation>
                            The ticket ID.
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>

Example A–16  getTicket_response.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="connectorCommon.xsd"/>
    <xsd:element name="getTicketResponse">
        <xsd:annotation>
            <xsd:documentation>
                The response from the external system upon a getTicketRequest request. It must contain a ticket ID from the output of the request Web Service.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ticketID" type="StringT128">
                    <xsd:annotation>
                        <xsd:documentation>
                            The ticket ID.
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
Example A–17  publishTicket.xsd

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
<xsd:include schemaLocation="connectorCommon.xsd"/>
<xsd:element name="publishTicketStatus">
    <xsd:annotation>
        <xsd:documentation>
            This section defines the request to publish ticket status from Ticketing
            system to EM when it is updated.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="ConnectorGUID" type="StringStrictT16">
                <xsd:annotation>
                    <xsd:documentation>
                        The GUID of the connector the request is to be sent to. The GUID
                        is communicated to the external system in the earlier requests to
                        create tickets. It is returned in the inbound data to associate
                        the date with the corresponding ticket.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="ticketID" type="StringT128">
                <xsd:annotation>
                    <xsd:documentation>
                        The ID of the ticket whose status is being updated.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="ticketStatus" type="StringT64">
                <xsd:annotation>
                    <xsd:documentation>
                        The new status of the ticket.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="RequestOccurTS" type="StringT64">
                <xsd:annotation>
                    <xsd:documentation>
                        Time when the inbound call is invoked.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```
This appendix provides all Management Connector specific error messages and debugging information. The errors are returned in the response model.

- Error Messages
- Debugging
  - Specifying the Debug Option
  - Viewing Debug Messages

## B.1 Error Messages

This section provides error codes, descriptions, causes, and suggested actions for all Connector Framework error messages.

**CNTR-0001**

**Cause:** Authentication failed. The credential to log in to Enterprise Manager is incorrect.

**Action:** Correct the Enterprise Manager credential element in the request.

**CNTR-0002**

**Cause:** In `setModel`, the requested aggregate target list is of size 0. This operation is not supported. In `setModel`, the requested aggregate target list is of size 0. This operation is not supported.

**Action:** Correct the request model to include two aggregate targets: one of type `cluster`, and the other of type `rac_database`.

**CNTR-0003**

**Cause:** The current cluster aggregate target and `rac_database` aggregate target have a different number of member targets.

**Action:** Make sure the numbers of members of the current cluster aggregate target and the `rac_database` aggregate target are the same.

**CNTR-0004**

**Cause:** The target name or type element is NULL in the request model for `getModel`.

**Action:** Correct the target name or target type in the request model.

**CNTR-0005**

**Cause:** There is an unrecognized property in the request model for `getModel`.

**Action:** Remove the unrecognized property.
CNTR-0006
Cause: The name or type of element of an aggregate target is NULL.
Action: Correct the name or type of the aggregate target in the request model.

CNTR-0007
Cause: The aggregate target type is something other than cluster and rac_database.
Action: Correct the aggregate target type. Only two types are supported in this release: cluster and rac_database.

CNTR-0008
Cause: The request model is invalid.
Action: Correct the request model. Make sure the request model has either zero or two aggregate targets (one of type cluster and one of type rac_database). If aggregate targets are included, make sure the numbers of targets in the two aggregate targets are the same with the same set of hosts. Make sure the name and host elements of each target in the cluster aggregate target are the same.

CNTR-0009
Cause: Either the cluster aggregate target or the rac_database aggregate target is missing in the request model.
Action: Add the missing aggregate target.

CNTR-0010
Cause: No software library image was found based on the description of the model.
Action: Correct the name of the software library image, or make sure the image is available in the Enterprise Manager software library.

CNTR-0011
Cause: No existing node could be found to run some steps of the add node job.
Action: Correct the request model to make sure all existing nodes are specified correctly in the request model.

CNTR-0012
Cause: No credential was specified for the RAC nodes.
Action: Add the credentials for the RAC nodes.

CNTR-0013
Cause: The name of the member target of the rac_database aggregate target does not follow the <rac_name>_<instance_name> naming rule.
Action: Correct the name of the member target of the rac_database aggregate target.

CNTR-0014
Cause: The storage element was missing during the RAC creation request.
Action: Add the storage element in the request model.

CNTR-0015
Cause: The number of nodes in the request model is less than the number of nodes in the current model minus one.
Action: Correct the response model by deleting only one node.

CNTR-0016
Cause: The cluster aggregate target and rac_database aggregate target have a different number of member targets.
Action: Correct the request model with the correct member targets for both the cluster aggregate target and rac_database aggregate target.

CNTR-0017
Cause: More than one node was specified in the request when the RAC database did not exist yet.
Action: Correct the request model to use only one node for the new RAC database.

CNTR-0018
Cause: The member targets of the aggregate targets do not match those in the Enterprise Manager repository.
Action: Correct the request model with the correct member target names.

CNTR-0019
Cause: Nothing to do: there are no member differences between the current and requested model. The members of the current model inside Enterprise Manager are the same as the one in the request. The connector cannot infer any action.
Action: Correct the request model to indicate a provisioning action.

CNTR-0020
Cause: The number of nodes in the request model is more than the number of nodes in the current model plus one.
Action: Correct the request model by adding only one node.

CNTR-0022
Cause: There is an error in the host of RAC aggregate target of the request model. The host attribute of the member targets does not match the host attribute in the Enterprise Manager repository.
Action: Correct the host attribute of the member target of the rac_database aggregate target.

CNTR-0023
Cause: There is an error in the host of the Oracle Clusterware aggregate target of the request model. The host attribute of the member targets does not match the host attribute in the Enterprise Manager repository.
Action: Correct the host attribute of the member target of the cluster aggregate target.

CNTR-0024
Cause: The host attribute of the member target of the cluster aggregate target does not match the host attribute for the corresponding member target of the rac_database aggregate target.
Action: Correct the host attribute of the member target of the cluster and rac_database aggregate targets.

CNTR-0025 (Windows only)
Cause: The sl_OHPartitionsAndSpace_valueFromDlg property is missing from the cluster aggregate target properties.

Action: Add the sl_OHPartitionsAndSpace_valueFromDlg property to the cluster aggregate target properties in the request model.

CNTR-0026 (Windows only)

Cause: The ret_PrivIntrList property is missing from the cluster aggregate target properties.

Action: Add the ret_PrivIntrList property to the cluster aggregate target properties in the request model.

B.2 Debugging

The Connector Framework uses the log4j logging utility to log the types of messages shown in Table B–1:

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Code Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning</td>
<td>WARN</td>
</tr>
<tr>
<td>Error</td>
<td>ERROR</td>
</tr>
<tr>
<td>Debugging</td>
<td>DEBUG</td>
</tr>
<tr>
<td>Information</td>
<td>INFO</td>
</tr>
</tbody>
</table>

B.2.1 Specifying the Debug Option

The following example shows the insertion of DEBUG in the following file:

$ORACLE_HOME/sysman/config/emomslogging.properties

Use the following emctl command to set the debug level as shown below:

emctl set property -name log4j.rootCategory -value "DEBUG, emlogAppender, emtrcAppender" -module emoms

Enter the SYSMAN password when prompted.

B.2.2 Viewing Debug Messages

The debug messages from the Connector Framework are displayed in the following file:

$INSTANCE_HOME/sysman/log/emoms.trc
This appendix provides the following sample implementations for a SCOM 2012 event connector:

<table>
<thead>
<tr>
<th>Sample Implementation Files (.xml and .xsl):</th>
<th>Sample Schema Files (.xsd):</th>
</tr>
</thead>
<tbody>
<tr>
<td>• connectorDeploy.xml</td>
<td>• connectorDeploy.xsd</td>
</tr>
<tr>
<td>• setup_request.xml</td>
<td>• EMEvent.xsd</td>
</tr>
<tr>
<td>• setup_response.xsl</td>
<td>• connectorCommon.xsd</td>
</tr>
<tr>
<td>• cleanup_request.xml</td>
<td>• EMEventResponse.xsd</td>
</tr>
<tr>
<td>• createEvent_request_2012.xsl</td>
<td>• externalEvent.xsd</td>
</tr>
<tr>
<td>• createEvent_response.xsl</td>
<td>• setupResponse.xsd</td>
</tr>
<tr>
<td>• updateEvent_request_2012.xsl</td>
<td>• initialize_response.xsd</td>
</tr>
<tr>
<td>• updateEvent_response.xsl</td>
<td>• uninitialize_response.xsd</td>
</tr>
</tbody>
</table>

### Example C–1 connectorDeploy.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ManagementConnector xmlns="http://xmlns.oracle.com/sysman/connector">
  <Name>SCOM 2012 Connector</Name>
  <Version>12.1.0.1.0</Version>
  <EMCompatibleVersion>12.1.0.1.0</EMCompatibleVersion>
  <Description>Microsoft System Center Operations Manager 2012 Integration with Enterprise Manager</Description>
  <Category>EventConnector</Category>
  <NewTargetType>
    <TargetTypeName>scom_managed_host</TargetTypeName>
    <TargetTypeDisplayName>SCOM Managed Host</TargetTypeDisplayName>
    <DefaultTargetName>generic_scom_managed_host</DefaultTargetName>
    <DefaultTargetDisplayName>Generic SCOM Managed Host</DefaultTargetDisplayName>
  </NewTargetType>
  <HTTPBasicAuthentication>
    <Username required="true">
      <VariableName>Username</VariableName>
      <DisplayName>SCOM Web Service Username</DisplayName>
    </Username>
    <Password required="true">
      <VariableName>Password</VariableName>
      <DisplayName>SCOM Web Service Password</DisplayName>
    </Password>
  </HTTPBasicAuthentication>
</ManagementConnector>
```
<Method>setup</Method>
<WebServiceEndpoint>
  <![CDATA[http://[Hostname]:8080/services/SCOM/SCOMService]]>
</WebServiceEndpoint>
<SOAPAction>setup</SOAPAction>
<SOAPBindingType>SOAP11HTTP_BINDING</SOAPBindingType>
</Service>

<Service>
  <Method>initialize</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[Hostname]:8080/services/SCOM/SCOMService]]>
  </WebServiceEndpoint>
  <SOAPAction>initialize</SOAPAction>
  <SOAPBindingType>SOAP11HTTP_BINDING</SOAPBindingType>
</Service>

<Service>
  <Method>createEvent</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[Hostname]:8080/services/SCOM/EventService]]>
  </WebServiceEndpoint>
  <SOAPAction>createEvent</SOAPAction>
  <SOAPBindingType>SOAP11HTTP_BINDING</SOAPBindingType>
</Service>

<Service>
  <Method>updateEvent</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[Hostname]:8080/services/SCOM/EventService]]>
  </WebServiceEndpoint>
  <SOAPAction>updateEvent</SOAPAction>
  <SOAPBindingType>SOAP11HTTP_BINDING</SOAPBindingType>
</Service>

<Service>
  <Method>uninitialize</Method>
  <WebServiceEndpoint>
    <![CDATA[http://[Hostname]:8080/services/SCOM/SCOMService]]>
  </WebServiceEndpoint>
  <SOAPAction>uninitialize</SOAPAction>
  <SOAPBindingType>SOAP11HTTP_BINDING</SOAPBindingType>
</Service>

<TemplateRegistration>
  <FileName>setup_request.xml</FileName>
  <InternalName>setup</InternalName>
  <TemplateName>Setup Request</TemplateName>
  <TemplateType>OutboundXML</TemplateType>
  <Description>This is the request xml file for the setup method</Description>
</TemplateRegistration>

<TemplateRegistration>
  <FileName>setup_response.xsl</FileName>
  <InternalName>setup</InternalName>
  <TemplateName>Setup Response</TemplateName>
  <TemplateType>InboundXSL</TemplateType>
  <Description>This is the response xsl file for the setup method</Description>
</TemplateRegistration>

<TemplateRegistration>
  <FileName>setup_request.xml</FileName>
  <InternalName>initialize</InternalName>
  <TemplateName>Initialize Request</TemplateName>
  <TemplateType>OutboundXML</TemplateType>
  <Description>This is the request xml file for the initialize method</Description>
</TemplateRegistration>
Example C–2  setup_request.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<register xmlns="http://oracle.com/services/adapter-framework"/>
```

Example C–3  setup_response.xsl

```xml
<?xml version='1.0' ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="'/">
  <SetupResponse xmlns="http://xmlns.oracle.com/sysman/connector">
    <ConnectorVariable>
      <VariableName>REGISTRATION_ID</VariableName>
      <VariableValue>Dummy</VariableValue>
    </ConnectorVariable>
  </SetupResponse>
</xsl:template>
```
Example C–4  cleanup_request.xml

<?xml version="1.0" encoding="UTF-8" ?>
<oracleaf:deregister
xmlns:oracleaf="http://oracle.com/services/adapter-framework">
  <deleteSubscriptions>true</deleteSubscriptions>
</oracleaf:deregister>

Example C–5  createEvent_request_2012.xsl

<?xml version='1.0' ?>
<xsl:stylesheet version='2.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
xmlns:a='http://xmlns.oracle.com/sysman/connector'>

<xsl:variable name='pad'><xsl:text>
</xsl:text></xsl:variable>

<xsl:template match='a:EMEvent'>
  <oracleaf:create
xmlns:oracleaf="http://oracle.com/services/adapter-framework">
    <event>
      <xsl:variable name='newLine'><xsl:text>
</xsl:text></xsl:variable>

<!-- SCOM alert description variables -->

<xsl:choose>
  <xsl:when test='normalize-space(a:SystemAttributes/a:OccurredDate) != '''>
    <xsl:value-of
select='substring-before(translate(a:SystemAttributes/a:OccurredDate, 'T', ' '), 
'.')'/>
  </xsl:when>
  <xsl:otherwise>N/A</xsl:otherwise>
</xsl:choose>

<xsl:variable name='fmtOccurDate'>Occurred Date: <xsl:value-of
select='$occurDate'/></xsl:variable>

<xsl:variable name='fmtReportDate'>Reported Date: <xsl:value-of
select='$reportDate'/></xsl:variable>

<xsl:variable name='fmtEventClass'>Event Class: <xsl:value-of
select='$eventClass'/></xsl:variable>

<xsl:variable name='fmtEventName'>Event Name: <xsl:value-of
select='$eventName'/></xsl:variable>
</xsl:template>
</xsl:stylesheet>
select="a:SystemAttributes/a:EventName"/>
</xsl:variable>
    <xsl:variable name="fmtTargetType">Target Type: <xsl:value-of
    select="a:SystemAttributes/a:SourceInfo/a:TargetInfo/a:TargetType"/>
</xsl:variable>
    <xsl:variable name="fmtTargetName">Target Name: <xsl:value-of
    select="a:SystemAttributes/a:SourceInfo/a:TargetInfo/a:TargetName"/>
</xsl:variable>
    <xsl:variable name="fmtSeverity">Severity: <xsl:value-of
    select="a:SystemAttributes/a:Severity"/>
</xsl:variable>
    <xsl:variable name="fmtMessage">Message: <xsl:value-of
    select="a:SystemAttributes/a:Message"/>
</xsl:variable>
    <xsl:variable name="fmtUrl">Event URL: <xsl:value-of
    select="a:SystemAttributes/a:EventURL"/>
</xsl:variable>

<!-- SCOM alert description -->
<description><xsl:text>Received event reported by Oracle Enterprise
Manager: </xsl:text></description>

<xsl:for-each select="a:SystemAttributes/a:SourceInfo/a:TargetInfo/a:TargetProperty">
    <xsl:if test="position() = 1">
        <xsl:value-of select="$newLine"/>
    </xsl:if>
    <xsl:value-of select="$newLine"/>
</xsl:for-each>

<xsl:for-each select="a:EventContextAttributes">
    <xsl:if test="position() = 1">
        <xsl:value-of select="$newLine"/>
    </xsl:if>
    <xsl:choose>
        <xsl:when test="a:StringAttribute">
            <xsl:value-of select="$newLine"/>
        </xsl:when>
        <xsl:when test="a:NumberAttribute">
            <xsl:value-of select="$newLine"/>
        </xsl:when>
    </xsl:choose>
</xsl:for-each>
<!-- SCOM alert name -->
<summary>
<xsl:value-of select="a:SystemAttributes/a:EventName"/>
</summary>

<!-- SCOM alert severity -->
<severity>
<xsl:choose>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'CLEAR'">Information</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'INFORMATIONAL'">Information</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'WARNING'">Warning</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'MINOR_WARNING'">Warning</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'CRITICAL'">Error</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'FATAL'">Error</xsl:when>
<xsl:otherwise>Error</xsl:otherwise>
</xsl:choose>
</severity>

<!-- SCOM alert priority -->
<priority>
<xsl:choose>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'CLEAR'">Low</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'INFORMATIONAL'">Low</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'WARNING'">Normal</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'MINOR_WARNING'">Normal</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'CRITICAL'">High</xsl:when>
<xsl:when test="a:SystemAttributes/a:SeverityCode = 'FATAL'">High</xsl:when>
<xsl:otherwise>Normal</xsl:otherwise>
</xsl:choose>
</priority>

<!-- SCOM history log variables -->
<xsl:variable name="tab"><xsl:text>    </xsl:text></xsl:variable>
<xsl:variable name="paddedMessage">
<xsl:call-template name="dopad">
<xsl:with-param name="pText" select="$fmtMessage"/>
</xsl:call-template>
</xsl:variable>
<xsl:variable name="urlpadlen">
<xsl:value-of select="(ceiling(string-length($fmtUrl) div 85)) * 85"/>
</xsl:variable>
<xsl:variable name="targetProps">
<xsl:for-each select="a:SystemAttributes/a:SourceInfo/a:TargetInfo/a:TargetProperty"/>
<xsl:variable name="targetPropsHeader">Target Properties Header</xsl:variable>
<xsl:variable name="contextHeader">Event Context Attributes</xsl:variable>

<xsl:if test="position() = 1">
  <xsl:value-of select="substring(concat($pad,$targetPropsHeader,$pad),1,170)"/>
</xsl:if>

<xsl:for-each select="a:Event">
  <xsl:if test="position() = 1">
    <xsl:value-of select="substring(concat($tab,./a:Name,$colon,./a:Value,$pad),1,85)"/>
  </xsl:if>
</xsl:for-each>

<xsl:variable name="contextAttr">
  <xsl:for-each select="a:EventContextAttributes">
    <xsl:if test="position() = 1">
      <xsl:value-of select="substring(concat($pad,$contextHeader,$pad),1,170)"/>
    </xsl:if>
    <xsl:choose>
      <xsl:when test="a:StringAttribute">
        <xsl:value-of select="substring(concat($tab,a:StringAttribute/a:Name,$colon,a:StringAttribute/a:Value,$pad),1,85)"/>
      </xsl:when>
      <xsl:when test="a:NumberAttribute">
        <xsl:value-of select="substring(concat($tab,a:NumberAttribute/a:Name,$colon,a:NumberAttribute/a:Value,$pad),1,85)"/>
      </xsl:when>
    </xsl:choose>
  </xsl:for-each>
</xsl:variable>

<xsl:variable name="history">
  <xsl:value-of select="substring(concat('Oracle Enterprise Manager created an event with the following attributes:', $pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtSeverity,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtReportDate,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtOccurDate,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtTargetName,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtTargetType,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtEventClass,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtEventName,$pad),1,85)"/>
  <xsl:value-of select="substring(concat($fmtUrl,$pad),1,$urlpadlen)"/>
  <xsl:value-of select="substring(concat($targetProps),1,85)"/>
  <xsl:value-of select="substring(concat($contextAttr),1,85)"/>
</xsl:variable>

<!-- SCOM history log information -->
<logs>
  <log>
    <description><xsl:value-of select="$history"/></description>
  </log>
</logs>

<!-- SCOM alert custom fields -->
<!-- Uncomment fields to be set and replace "VALUE" with the actual value -->
<!--
  <string-field name="CustomField1">VALUE</string-field>
  <string-field name="CustomField2">VALUE</string-field>
-->
<string-field name="CustomField3">VALUE</string-field>
<string-field name="CustomField4">VALUE</string-field>
<string-field name="CustomField5">VALUE</string-field>
<string-field name="CustomField6">VALUE</string-field>
<string-field name="CustomField7">VALUE</string-field>
<string-field name="CustomField8">VALUE</string-field>
<string-field name="CustomField9">VALUE</string-field>
<string-field name="CustomField10">VALUE</string-field>

<xsl:template name="dopad">
<xsl:param name="pText" />
<xsl:param name="pDelim" select="" "/>

<xsl:if test="string-length($pText) &gt; 0">
<xsl:variable name="str" select="substring($pText,1,85)"/>
<xsl:variable name="line" select="substring(concat($line,$pad),1,85)"/>

<xsl:choose>
  <xsl:when test="contains($str,' ')">
    <xsl:call-template name="splitLine">
      <xsl:with-param name="pText" select="$str"/>
      <xsl:with-param name="pDelim" select="$pDelim"/>
    </xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="$str"/>
  </xsl:otherwise>
</xsl:choose>
</xsl:variable>

<xsl:value-of select="substring(concat($line,$pad),1,85)"/>
<xsl:variable name="remstr" select="substring($pText,string-length($line)+1)"/>

<xsl:call-template name="dopad">
  <xsl:with-param name="pText" select="$remstr"/>
  <xsl:with-param name="pDelim" select="$pDelim"/>
</xsl:call-template>
</xsl:if>
</xsl:template>

<xsl:template name="splitLine">
<xsl:param name="pText" />
<xsl:param name="pDelim" select="" "/>

<xsl:if test="contains($pText, $pDelim)">
<xsl:value-of select="substring-before($pText, $pDelim)"/>
<xsl:call-template name="splitLine">
  <xsl:with-param name="pText" select="substring-after($pText, $pDelim)"/>
</xsl:call-template>
</xsl:if>
</xsl:template>
**Example C–6  createEvent_response.xsl**

```xml
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:oracleaf="http://oracle.com/services/adapter-framework"
xmlns:a="http://xmlns.oracle.com/sysman/connector">

<xsl:template match="oracleaf:createResponse/return">
  <a:EMEventResponse>
    <xsl:choose>
      <xsl:when test="identifier">
        <a:SuccessFlag>true</a:SuccessFlag>
        <a:ExternalEventId>
          <xsl:value-of select="identifier"/>
        </a:ExternalEventId>
      </xsl:when>
      <xsl:otherwise>
        <a:SuccessFlag>false</a:SuccessFlag>
        <a:ErrorMessage>Request to create an event in SCOM failed</a:ErrorMessage>
      </xsl:otherwise>
    </xsl:choose>
  </a:EMEventResponse>
</xsl:template>

</xsl:stylesheet>
```

**Example C–7  updateEvent_request_2012.xsl**

```xml
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:a="http://xmlns.oracle.com/sysman/connector">

<xsl:variable name="pad"><xsl:text>  </xsl:text></xsl:variable>

<xsl:template match="a:EMEvent">
  <oracleaf:update
xmlns:oracleaf="http://oracle.com/services/adapter-framework">
    <event>
      <!-- SCOM alert GUID to update -->
      <identifier>
        <xsl:value-of select="a:ExternalEventID"/>
      </identifier>

      <!-- SCOM alert resolution state -->
      <status>
        <xsl:choose>
          <xsl:when test="a:SystemAttributes/a:SeverityCode = 'CLEAR'">255</xsl:when>
          <xsl:otherwise>0</xsl:otherwise>
        </xsl:choose>
      </status>

      <!-- SCOM history log variables -->
      <xsl:variable name="reportDate">
        <xsl:choose>
          <xsl:when test="normalize-space(a:SystemAttributes/a:ReportedDate) != '...'">
            <xsl:value-of
```
select="substring-before(translate(a:SystemAttributes/a:ReportedDate, 'T', ' '), '.')"/>
   </xsl:when>
   <xsl:otherwise>N/A</xsl:otherwise>
 </xsl:choose>
</xsl:variable>

<xsl:variable name="_title">
 <xsl:choose>
   <xsl:when test="a:SystemAttributes/a:SeverityCode = 'CLEAR'>Oracle Enterprise Manager cleared event<xsl:value-of select="$pad"/></xsl:when>
   <xsl:when test="a:SystemAttributes/a:SeverityCode = 'WARNING'>Oracle Enterprise Manager changed event severity to Warning<xsl:value-of select="$pad"/></xsl:when>
 </xsl:choose>
</xsl:variable>

<xsl:variable name="_reportDate">Reported Date: <xsl:value-of select="$_reportDate"/></xsl:variable>

<xsl:variable name="paddedMessage">
 <xsl:call-template name="dopad">
   <xsl:with-param name="pText" select="$_message"/>
 </xsl:call-template>
</xsl:variable>

<xsl:variable name="_history">
 <xsl:value-of select="substring($_title,1,85)"/>
 <xsl:value-of select="substring($_reportDate,1,85)"/>
 <xsl:value-of select="$paddedMessage"/>
 <xsl:value-of select="substring($_severity,1,85)"/>
</xsl:variable>

<!-- SCOM history log information -->
<logs>
 <log>
   <description><xsl:value-of select="$_history"/></description>
 </log>
</logs>

<extended-fields>
 <!-- SCOM alert custom fields -->
 <!-- Uncomment fields to be set and replace "VALUE" with the actual value -->
 <!--
 <string-field name="CustomField1">VALUE</string-field>
 <string-field name="CustomField2">VALUE</string-field>
 <string-field name="CustomField3">VALUE</string-field>
 <string-field name="CustomField4">VALUE</string-field>
 <string-field name="CustomField5">VALUE</string-field>
 <string-field name="CustomField6">VALUE</string-field>
 <string-field name="CustomField7">VALUE</string-field>
 <string-field name="CustomField8">VALUE</string-field>
 -->
Example C–8 updateEvent_request_2012_alt.xsl

<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    <string-field name="CustomField9">VALUE</string-field>
    <string-field name="CustomField10">VALUE</string-field>
</xsl:stylesheet>
<xsl:variable name="pad"><xsl:text>
</xsl:text></xsl:variable>
<xsl:variable name="syncHistory">true</xsl:variable>
<xsl:template match="a:EMEvent">
  <xsl:choose>
    <xsl:when test="a:SystemAttributes/a:SeverityCode = 'CLEAR'">
      <!-- The event has been cleared. Close the alert in SCOM -->
      <oracleaf:update xmlns:oracleaf="http://oracle.com/services/adapter-framework">
        <event>
          <!-- SCOM alert GUID to update -->
          <identifier>
            <xsl:value-of select="a:ExternalEventID"></xsl:value-of>
          </identifier>
          <!-- SCOM alert resolution state -->
          <status>255</status>
          <!-- SCOM history log information -->
          <logs>
            <log>
              <description>Oracle Enterprise Manager cleared alert</description>
            </log>
          </logs>
          <extended-fields>
            <string-field name="CloseAssociatedAlerts"><xsl:value-of select="a:ExternalEventID"></xsl:value-of></string-field>
            <!-- SCOM alert custom fields -->
            <!-- Uncomment fields to be set and replace "VALUE" with the actual value -->
            <!-- <string-field name="CustomField1">VALUE</string-field> -->
            <!-- <string-field name="CustomField2">VALUE</string-field> -->
            <!-- <string-field name="CustomField3">VALUE</string-field> -->
            <!-- <string-field name="CustomField4">VALUE</string-field> -->
            <!-- <string-field name="CustomField5">VALUE</string-field> -->
            <!-- <string-field name="CustomField6">VALUE</string-field> -->
            <!-- <string-field name="CustomField7">VALUE</string-field> -->
            <!-- <string-field name="CustomField8">VALUE</string-field> -->
            <!-- <string-field name="CustomField9">VALUE</string-field> -->
            <!-- <string-field name="CustomField10">VALUE</string-field> -->
          </extended-fields>
        </event>
      </oracleaf:update>
    </xsl:when>
    <xsl:otherwise>
      <oracleaf:create xmlns:oracleaf="http://oracle.com/services/adapter-framework">
        <event>
          <xsl:variable name="newLine"><xsl:text>
</xsl:text></xsl:variable>
</xsl:variable>
</xsl:otherwise>
</oracleaf:create>
</event>
</xsl:when>
</xsl:choose>
</xsl:template>
<!-- SCOM alert description variables -->
<xsl:variable name="occurDate">
  <xsl:choose>
    <xsl:when test="normalize-space(a:SystemAttributes/a:OccurredDate) != ''">
      <xsl:value-of select="substring-before(translate(a:SystemAttributes/a:OccurredDate, 'T', ' '), '.')"/>
    </xsl:when>
    <xsl:otherwise>N/A</xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<xsl:variable name="reportDate">
  <xsl:choose>
    <xsl:when test="normalize-space(a:SystemAttributes/a:ReportedDate) != ''">
      <xsl:value-of select="substring-before(translate(a:SystemAttributes/a:ReportedDate, 'T', ' '), '.')"/>
    </xsl:when>
    <xsl:otherwise>N/A</xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<xsl:variable name="fmtOccurDate">Occurred Date: <xsl:value-of select="$occurDate"/></xsl:variable>

<xsl:variable name="fmtReportDate">Reported Date: <xsl:value-of select="$reportDate"/></xsl:variable>


<xsl:variable name="fmtEventName">Event Name: <xsl:value-of select="a:SystemAttributes/a:EventName"/></xsl:variable>

<xsl:variable name="fmtTargetType">Target Type: <xsl:value-of select="a:SystemAttributes/a:SourceInfo/a:TargetInfo/a:TargetType"/></xsl:variable>

<xsl:variable name="fmtTargetName">Target Name: <xsl:value-of select="a:SystemAttributes/a:SourceInfo/a:TargetInfo/a:TargetName"/></xsl:variable>


<!-- SCOM alert description -->
<description><xsl:text>Received event reported by Oracle Enterprise Manager:</xsl:text>

<xsl:value-of select="$newLine"/>
<xsl:value-of select="$fmtOccurDate"/>
<xsl:value-of select="$fmtReportDate"/>
<xsl:value-of select="$fmtEventClass"/>
</description>
Example C-9  updateEvent_response.xsl

```xml
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:oracleaf="http://oracle.com/services/adapter-framework"
    xmlns:a="http://xmlns.oracle.com/sysman/connector">
    <xsl:template match="/return">
        <a:EMEventResponse>
            <xsl:choose>
                <xsl:when test="identifier">
                    <a:SuccessFlag>true</a:SuccessFlag>
                    <a:ExternalEventId>
                        <xsl:value-of select="identifier"/>
                    </a:ExternalEventId>
                </xsl:when>
                <xsl:otherwise>
                    <a:SuccessFlag>false</a:SuccessFlag>
                    <a:ErrorMessage>Request to update an event in SCOM failed</a:ErrorMessage>
                </xsl:otherwise>
            </xsl:choose>
        </a:EMEventResponse>
    </xsl:template>
</xsl:stylesheet>
```

Example C-10  connectorDeploy.xsd

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="connectorCommon.xsd"/>
    <xsd:element name="ManagementConnector">
        <xsd:annotation>
            <xsd:documentation>Deployment Descriptor for Management Connectors</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
```

```xml
```
<xsd:element name='Name' type='StringT64'>
  <xsd:annotation>
    <xsd:documentation>
      The name of the connector type.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name='Version' type='VersionT'>
  <xsd:annotation>
    <xsd:documentation>
      Version of the connector type.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name='EMCompatibleVersion' type='VersionT'>
  <xsd:annotation>
    <xsd:documentation>
      The EM compability version of the connector type.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name='Description' type='StringT256'>
  <xsd:annotation>
    <xsd:documentation>
      The description of the connector type.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name='Category'>
  <xsd:annotation>
    <xsd:documentation>
      The category of the connector type. It must be one of the three values listed next.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base='xsd:string'>
      <xsd:enumeration value='EventConnector'/>
      <xsd:enumeration value='TicketingConnector'/>
      <xsd:enumeration value='ChangeManagementConnector'/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<!-- NewTargetType is for EventConnector only. -->
<xsd:element name='NewTargetType' minOccurs='0'>
  <xsd:annotation>
    <xsd:documentation>
      New target type definition for event connectors. This target type will be registered with Enterprise Manager and target instances can be created subsequently, including a default target. These targets are used to accommodate external events.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name='TargetTypeName' type='StringStrictT64'>
        <xsd:annotation>
          <xsd:documentation>
            The name of the target type.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="TargetTypeDisplayName" type="StringT128">
  <xsd:annotation>
    <xsd:documentation>
The name of the target type, as shown on UI.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="DefaultTargetName" type="StringStrictT256">
  <xsd:annotation>
    <xsd:documentation>The name of the default target of the target type. The default target will be used as a generic bucket to hold external events.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="DefaultTargetDisplayName" type="StringT256">
  <xsd:annotation>
    <xsd:documentation>The name of the default target of the target type, to be displayed on UI.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="SOAPHeaderAuthentication" type="SOAPHeaderAuthenticationType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Specification for SOAP Header authentication.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="HTTPBasicAuthentication" type="UsernamePasswordAuthenticationType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Specification for HTTP basic authentication.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="UserNameTokenAuthentication" type="UsernamePasswordAuthenticationType" minOccurs="0">
  <xsd:annotation>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="ConfigVariable" type="ConfigVariableType" minOccurs="0" maxOccurs="20">
  <xsd:annotation>
    <xsd:documentation>
    </xsd:annotation>
</xsd:element>
The variables used during connector configuration. These variables are required by external system to complete connector configuration, which includes registering with the external system. For instance, one configuration variable can be the resolution state required by Microsoft Operation Manager.

An optional variable used to test connection to an external server.

Specification for web services, which define how connector framework can communicate with external system.

Specification for the URL link to the external server, including the URL pattern and server specific variables. It is used to provide links to external server for viewing ticket details.

Specification for template registration. A template is registered based on the information provided in the element. A connector deployment descriptor can have an optional list of upto 50 template registration elements.
<xsd:element name="Method">
  <xsd:annotation>
    <xsd:documentation>
The name of the web service method. Each connector category has a predefined set of methods as defined next.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <!-- event connector: -->
      <xsd:enumeration value="setup"/>
      <xsd:enumeration value="initialize"/>
      <xsd:enumeration value="getNewAlerts"/>
      <xsd:enumeration value="getUpdatedAlerts"/>
      <xsd:enumeration value="acknowledgeAlerts"/>
      <xsd:enumeration value="updateAlerts"/>
      <xsd:enumeration value="createEvent"/>
      <xsd:enumeration value="updateEvent"/>
      <xsd:enumeration value="uninitialize"/>
      <xsd:enumeration value="cleanup"/>
      <!-- ticketing connector: -->
      <xsd:enumeration value="createTicket"/>
      <xsd:enumeration value="updateTicket"/>
      <xsd:enumeration value="getTicket"/>
      <!-- change management connector: -->
      <xsd:enumeration value="publishCS"/>
      <xsd:enumeration value="updateChangeRequest"/>
      <xsd:enumeration value="getChangeRequest"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="WebServiceEndpoint" type="StringT256">
  <xsd:annotation>
    <xsd:documentation>
The web service end point indicating a specific location for accessing a service.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="SOAPAction" type="StringT64" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
The SOAP action which carries out the web service call for the method.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="SOAPBindingType" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
The type of SOAP over HTTP binding. Choose from one of the four options defined next.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="SOAP11HTTP_BINDING"/>
      <xsd:enumeration value="SOAP12HTTP_BINDING"/>
      <xsd:enumeration value="SOAP11HTTP_MTOM_BINDING"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="Username" type="ConfigVariableType">
    <xsd:annotation>
        <xsd:documentation>The username of the authentication.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="Password" type="ConfigVariableType">
    <xsd:annotation>
        <xsd:documentation>The password of the authentication.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="AuthVariable" type="ConfigVariableType" minOccurs="0" maxOccurs="20">
    <xsd:annotation>
        <xsd:documentation>An optional list of extra authentication variables besides username and password.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="SOAPHeader" type="StringT256">
    <xsd:annotation>
        <xsd:documentation>A SOAP header string serving as template for the SOAP header. It is to be updated with user inputs for variables defined above and bound with a HTTP request.</xsd:documentation>
    </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:complexType>
</xsd:complexType>
</xsd:complexType>
<xsd:complexType name="ConfigVariableType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for configuration variables.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="VariableName" type="StringStrictT32">
      <xsd:annotation>
        <xsd:documentation>
          Name of the variable.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="DisplayName" type="StringT64">
      <xsd:annotation>
        <xsd:documentation>
          Name of the variable used for display on UI.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:attribute name="required" type="xsd:boolean" default="false">
      <xsd:annotation>
        <xsd:documentation>
          A Flag indicating whether or not the variable is mandatory.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExternalURLType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for external URL.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="Pattern" type="StringT256">
      <xsd:annotation>
        <xsd:documentation>
          The URL pattern used to format links to the external server.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="ConfigVariable" type="ConfigVariableType" minOccurs="0" maxOccurs="50">
      <xsd:annotation>
        <xsd:documentation>
          An optional list of configuration variablees representing the
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
details
    of the external server. They are used for constructing links to
    the server based on the URL pattern.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TemplateRegistrationType">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a complex type for template registration metadata
            which is used to register templates during connector deployment.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="FileName" type="StringT256">
            <xsd:annotation>
                <xsd:documentation>
                    The template file name.
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="InternalName" type="StringStrictT128">
            <xsd:annotation>
                <xsd:documentation>
                    A name representing the template in the connector framework.
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="TemplateName" type="StringStrictT128">
            <xsd:annotation>
                <xsd:documentation>
                    The template display name to be used on UI.
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:element name="TemplateType">
            <xsd:annotation>
                <xsd:documentation>
                    The template type as one of the three options defined next.
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="InboundXSL"/>
                    <xsd:enumeration value="OutboundXSL"/>
                    <xsd:enumeration value="OutboundXML"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Description" type="StringT512">
            <xsd:annotation>
                <xsd:documentation>
                    A description of the template.
                </xsd:documentation>
            </xsd:annotation>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
Example C-11  EMEvent.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
  <xsd:include schemaLocation="connectorCommon.xsd"/>
  <xsd:element name="EMEvent">
    <xsd:annotation>
      <xsd:documentation>
        This section defines an EM event made available through the connector framework.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ConnectorGUID" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              A unique ID to identify the connector used to forward the EM event to the targeted external event system.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="ExternalEventID" type="xsd:string" minOccurs="0">
          <xsd:annotation>
            <xsd:documentation>
              The ID to identify the event created in the external event system.
              It is generated in the external system and used to update the external event.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="NotificationRuleOwner" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              The owner of the notification rule which delivers the event.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="NotificationRuleName" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              The name of the notification rule which delivers the event.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="ConnectorVariable" type="VariableType" minOccurs="0" maxOccurs="50">
          <xsd:annotation>
            <xsd:documentation>
              An optional list of up to 50 connector variables that contain name/value pairs. They correspond to the ConfigVariable defined in connectorDeploy.xsd.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
<xsd:documentation>An optional list of up to 50 property variables as defined in connectorCommon.xsd.</xsd:documentation>
</xsd:element>
<xsd:element name="SystemAttributes" type="EventSystemAttributesType">
  <xsd:annotation>
    <xsd:documentation>A list of attributes for events as defined by EM event system.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="EventClassSpecificAttributes">
  <xsd:annotation>
    <xsd:documentation>A list of attributes for events that are specific to the event class.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="200">
        <xsd:element name="StringAttribute" type="StringValueType">
          <xsd:annotation>
            <xsd:documentation>A String attribute.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="NumberAttribute" type="StringValueType">
          <xsd:annotation>
            <xsd:documentation>A Number attribute.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="RawAttribute" type="StringValueType">
          <xsd:annotation>
            <xsd:documentation>An attribute of type Raw.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="DateAttribute" type="DateValueType">
          <xsd:annotation>
            <xsd:documentation>An attribute of type Date.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
Event Connector Samples

C-27

Event Context Attributes

A list of contextual attributes that is captured by the source system at the point of event generation that could be useful for diagnosis.

Event System Attributes Type

This section defines a complex type for system attributes provided by EM event system.

Event Class

The class of the event. For instance, target availability event, metric alert event, job status change event.

Event ID

The ID to identify a single event instance.

Sequence ID
<xsd:documentation>ID to identify a sequene of events which share the same event life</xsd:documentation>
</xsd:element>
<xsd:element name="OccurredDate" type="xsd:dateTime" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Date when the event occured.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="ReportedDate" type="xsd:dateTime">
  <xsd:annotation>
    <xsd:documentation>The date timestamp when the EM event publishing system is reporting the event.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="DisplayTZ" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>The display timezone region associated with the event. Event publishers can specify the time zone the event should be displayed in.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="EventName" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>Name of the event.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Severity" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>Severity level of the current event. The value changes based on local language setting.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SeverityCode" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>Internal Severity value of the current event.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="SourceInfo" type="SourceInfoType">
  <xsd:annotation>
    <xsd:documentation>The source information of the EM subsystems or components that raises the event.</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Message" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      A description of the event.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="ActionMessage" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      The action message for the event that helps diagnosing the issue.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="EventURL" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation>
      A URL of the event on EM incident console.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="AutoClose" type="xsd:boolean">
  <xsd:annotation>
    <xsd:documentation>
      A flag indicating if an event is auto closed by the system, or it has to be manually closed by users.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:element name="EventCategory" type="xsd:string" minOccurs="0" maxOccurs="50">
  <xsd:annotation>
    <xsd:documentation>
      An optional list of event categories to which the event belongs.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="StringValue_Type">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for a general name/value pair, both in String.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          The name of the String.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>

    <xsd:element name="Value" type="xsd:string">
      <xsd:annotation>
        
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
The value of the String.
</xsd:documentation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DateValueType">
  <xsd:annotation>
    <xsd:documentation>
      This section defines a complex type for a name/value pair, with name as a String name and value as a Date value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          The name of the Date.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="Value" type="xsd:dateTime">
      <xsd:annotation>
        <xsd:documentation>
          The value of the Date.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

Example C–12  connectorCommon.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://xmlns.oracle.com/sysman/connector"
  targetNamespace="http://xmlns.oracle.com/sysman/connector"
  elementFormDefault="qualified">
  <xsd:include schemaLocation="externalEvent.xsd"/>
  <xsd:complexType name="SourceInfoType">
    <xsd:annotation>
      <xsd:documentation>
        This section defines a complex type for Source Information.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="SourceObjInfo" type="SourceObjInfoType" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>
            This element defines the data structure for the source object, the EM subsystem or component, that raises an EM event or an incident.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="TargetInfo" type="TargetInfoType" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
The element defines the data structure for an EM target as related to the connector framework.

```xml
<xs:element name="SourceObjInfoType">
  <xs:annotation>
    <xs:documentation>
      This section defines a complex type for Source Object Information.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ObjID" type="xsd:string">
      <xs:annotation>
        <xs:documentation>
          The unique ID to identify the source object.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="ObjName" type="xsd:string">
      <xs:annotation>
        <xs:documentation>
          The name of the source object.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="ObjOwner" type="xsd:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>
          The owner of the source object.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="SourceObjType" type="xsd:string">
      <xs:annotation>
        <xs:documentation>
          The type of the source object.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="SourceObjSubType" type="xsd:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>
          The subtype of the source object.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

```xml
<xs:complexType name="TargetInfoType">
  <xs:annotation>
    <xs:documentation>
      This section defines a complex type for target information.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="TargetGUID" type="xsd:string">
      <xs:annotation>
        <xs:documentation>
          The unique ID to identify the target object.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```
<xsd:documentation>
A unique GUID for the target.
</xsd:documentation>
</xsd:element>
<xsd:element name="TargetName" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
Name of the target.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="TargetType" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
Type of the target.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="TargetTypeLabel" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
The display label of the target type.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="TargetURL" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
The URL of the target.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="TargetProperty" type="PropertyType" minOccurs="0" maxOccurs="50">
<xsd:annotation>
<xsd:documentation>
An optional list of properties for the target.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PropertyType">
<xsd:annotation>
<xsd:documentation>
This section defines a complex type for a property attribute.
</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
<xsd:element name="Name" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
A string name defining a property attribute.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="Value" type="xsd:string" nillable="true">
<xsd:annotation>
<xsd:documentation>
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
A non-null string value.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="VariableType">
<xsd:annotation>
<xsd:documentation>
This section defines a complex type for a general variable.
</xsd:documentation>
</xsd:annotation>
</xsd:sequence>
<xsd:element name="VariableName" type="StringStrictT32">
<xsd:annotation>
<xsd:documentation>
Name of the variable. It has to be a string containing 1 or upto 32 upper case or lower case letters or numbers.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="VariableValue" type="StringT2048">
<xsd:annotation>
<xsd:documentation>
Value of the variable. It has to be a string containing 1 or upto 2048 characters.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetAlertsResponse">
<xsd:annotation>
<xsd:documentation>
This section defines a complex type for responses to a getAlerts request.
</xsd:documentation>
</xsd:annotation>
</xsd:sequence>
<xsd:element name="Alert" minOccurs="0" maxOccurs="200">
<xsd:annotation>
<xsd:documentation>
The individual alerts contained in the response. A response may have upto 200 alerts.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:complexType>
<xsd:complexType name="GetAlertsResponse">
<xsd:annotation>
<xsd:documentation>
Details of the external event in the alert, as defined in ExternalEvent.xsd.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="InstanceVariable" type="VariableType" minOccurs="0" maxOccurs="50">
<xsd:annotation>
<xsd:documentation>

</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:documentation>
A list of instance variables for the alert.
</xsd:documentation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ConnectorVariablesType">
<xsd:annotation>
<xsd:documentation>
This section defines a complex type for connector variables. An element of type ConnectorVariablesType may have up to 50 connector variables, as defined next.
</xsd:documentation>
</xsd:annotation>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ConnectorVariable" type="VariableType" minOccurs="0" maxOccurs="50">
<xsd:annotation>
<xsd:documentation>
A connector variable as a name/value pair.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="StringT64">
<xsd:annotation>
<xsd:documentation>
This section defines a simple type for a String with maximum length of 64 bytes.
</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string">
<xsd:minLength value="1"/>
<xsd:maxLength value="64"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="StringT128">
<xsd:annotation>
<xsd:documentation>
This section defines a simple type for a String with maximum length of 128 bytes.
</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string">
<xsd:minLength value="1"/>
<xsd:maxLength value="128"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="StringT256">
<xsd:annotation>
<xsd:documentation>
This section defines a simple type for a String with maximum length of 256 bytes.
</xsd:documentation>
</xsd:annotation>
</xsd:simpleType>
<xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="256"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringT512">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 512 bytes.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="512"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringT2048">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 2048 bytes.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="2048"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringStrictT16">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 16 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="16"/>
        <xsd:pattern value="([a-zA-Z0-9_])*/"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringStrictT32">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 32 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="32"/>
        <xsd:pattern value="([a-zA-Z0-9_])*/"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StringStrictT64">
    <xsd:annotation>
        <xsd:documentation>
            This section defines a simple type for a String with maximum length of 64 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="64"/>
        <xsd:pattern value="([a-zA-Z0-9_])*/"/>
    </xsd:restriction>
</xsd:simpleType>
This section defines a simple type for a String with maximum length of 64 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.

This section defines a simple type for a String with maximum length of 128 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.

This section defines a simple type for a String with maximum length of 256 bytes. The String can only contain lower or upper case letters, numbers, and the underscore characters.

This section defines a simple type for a String with maximum length of 20 bytes. The String can only contain numbers and the period characters.
**Example C–13  EMEventResponse.xsd**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
  <xsd:include schemaLocation="connectorCommon.xsd"/>
  <xsd:element name="EMEventResponse">
    <xsd:annotation>
      <xsd:documentation>
        The response from external server for an EM event it has received.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SuccessFlag" type="xsd:boolean">
          <xsd:annotation>
            <xsd:documentation>
              The flag to indicate whether or not the event has been successfully inserted or updated at the external system.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:choice>
          <xsd:element name="ExternalEventId" type="StringT128">
            <xsd:annotation>
              <xsd:documentation>
                The ID to identify the event created in the external event system. It is returned by the external system when the event is successfully inserted or updated.
              </xsd:documentation>
            </xsd:annotation>
          </xsd:element>
          <xsd:element name="ErrorMessage" type="StringT2048">
            <xsd:annotation>
              <xsd:documentation>
                The error message returned by the external system when the event fails to be inserted or updated.
              </xsd:documentation>
            </xsd:annotation>
          </xsd:element>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

**Example C–14  externalEvent.xsd**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://xmlns.oracle.com/sysman/connector"
    targetNamespace="http://xmlns.oracle.com/sysman/connector"
    elementFormDefault="qualified">
  <xsd:element name="ExternalEvent">
    <xsd:annotation>
      <xsd:documentation>
        The response from external server for an EM event it has received.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SuccessFlag" type="xsd:boolean">
          <xsd:annotation>
            <xsd:documentation>
              The flag to indicate whether or not the event has been successfully inserted or updated at the external system.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="ExternalEventId" type="StringT128">
          <xsd:annotation>
            <xsd:documentation>
              The ID to identify the event created in the external event system. It is returned by the external system when the event is successfully inserted or updated.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="ErrorMessage" type="StringT2048">
          <xsd:annotation>
            <xsd:documentation>
              The error message returned by the external system when the event fails to be inserted or updated.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
This section defines the attribute requirement of an external event for the connector framework to process it.

```xml
<xs:complexType name="ExternalEventSystemAttributesType">
  <xs:documentation>This section defines a complex type for system attributes required for all external events.</xs:documentation>
  <xs:sequence>
    <xs:element name="eventName" type="xs:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Name of the event.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="category" type="xs:string" minOccurs="0" maxOccurs="50">
      <xs:annotation>
        <xs:documentation>The event category to which the event belongs.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="targetName" type="xs:string">
      <xs:annotation>
        <xs:documentation>Name of the target on which event was generated. It refers to an entity in external systems similar to an EM target.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="targetType" type="xs:string">
      <xs:annotation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```
<xsd:documentation>
The type of the target. Target types defined for event connectors are used. See connectorDeploy.xsd.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="occurrenceDate" type="xsd:dateTime">
<xsd:annotation>
<xsd:documentation>
Date when the event occurred.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="detectedDate" type="xsd:dateTime">
<xsd:annotation>
<xsd:documentation>
Date when the event was last detected.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="autoClose" type="xsd:boolean">
<xsd:annotation>
<xsd:documentation>
A flag indicating if an event is auto closed by the system, or it has to be manually closed by users.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="message" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
A description of the event.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="severity" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
Severity level of the event.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ExternalEventClassAttributesType">
<xsd:annotation>
<xsd:documentation>
This section defines a complex type for class specific attributes required for all external events in the class.
</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
<xsd:element name="external_event_id" type="xsd:string">
<xsd:annotation>
<xsd:documentation>
ID used in external system to identify the event.
</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="external_rule_id" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>Optional rule ID that delivered the event in the external system.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="external_host" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>Optional host information from external system where event was generated.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="external_source" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>Optional source information from the external system. </xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="external_severity" type="xsd:string">
    <xsd:annotation>
        <xsd:documentation>Severity level of the event on external system.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="external_status" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>Status of the event on external system.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field1" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>An optional field.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field2" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>An optional field.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field3" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>An optional field.</xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:element name="custom_field4" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>
    </xsd:annotation>
</xsd:element>
Example C–15  setupResponse.xsd

<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmllns='http://xmlns.oracle.com/sysman/connector'
targetNamespace='http://xmlns.oracle.com/sysman/connector'
elementFormDefault='qualified'>
  <xsd:include schemaLocation="connectorCommon.xsd"/>
  <xsd:element name="SetupResponse" type="ConnectorVariablesType">
    <xsd:annotation>
      <xsd:documentation>
        Response from external system for a setup request. It consists of a list of connector variables, i.e. name/value pairs.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:schema>

Example C–16  initialize_response.xsd

<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmllns='http://xmlns.oracle.com/sysman/connector'
targetNamespace='http://xmlns.oracle.com/sysman/connector'
elementFormDefault='qualified'>
  <xsd:include schemaLocation="connectorCommon.xsd"/>
  <xsd:element name="InitializeResponse" type="ConnectorVariablesType">
    <xsd:annotation>
      <xsd:documentation>
        The response for an initialize request. It contains a list of connector variables, which are name/value pairs.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:schema>

Example C–17  uninitialize_response.xsd

<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmllns='http://xmlns.oracle.com/sysman/connector'
targetNamespace='http://xmlns.oracle.com/sysman/connector'
elementFormDefault='qualified'>
<xsd:include schemaLocation="connectorCommon.xsd"/>
<xsd:element name="UninitializeResponse" type="ConnectorVariablesType">
  <xsd:annotation>
    <xsd:documentation>
      The response for an uninitialize request. It contains a list of
      connector variables, which are name/value pairs.
    </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:schema>
This appendix contains an example that shows how to create two default templates for a ticketing connector and shows the completed templates:

- Create Mappings
- Update Mappings
- Completed Templates
  - Completed First Template Example
  - Completed Second Template Example

Listed below are the steps that would be taken to create the templates.

1. Study data in external ticketing application.

   The first step is to study the web service for the external ticketing application and determine what fields are available and what the data looks like in those fields.

   Study of the web service WSDL shows that the root element for create requests is `createRequest` and the root element for update requests is `updateRequest`. The namespace for both is `http://samplecompany.com`. The WSDL also shows that there are four required fields on a create. These fields are `summary`, `description`, `severity`, and `assignment group`. The severity field must be a numeric value from 1 to 5. Where 1 is the highest and 5 is the lowest. We have decided to just populate the required fields on the create request.

   For the update, only the identifier field and a history log field are required. We will be specifying those fields along with the severity field on update requests.

2. Study data in Enterprise Manager.

   Now that we understand the data required to create/update tickets, we need to understand what data is available in Enterprise Manager and where it is located. Study of the data showed that all of the data that we need is located in the `SystemAttributes` element. There is a `Summary` element that has summary information and there is a `SeverityCode` field that can be used as well. Also included in the `SystemAttributes` element is target information that we would like to see in the ticket description. Some of the ticket fields will have to be hard coded because there is not sufficient information in the Enterprise Manager incident data.

3. Determine number of templates and the mapping for each.

   We have decided to create two default templates for our connector. One will leave the ticket open and update the history log whenever the incident in Enterprise Manager is cleared, and the other template will close the ticket.
The mappings to create the ticket are the same for both templates. Listed below are
the mappings that have been identified for the create operation.

- Summary will be set to the contents of the SystemAttributes/Summary field
- Description will be set to the contents of the SystemAttributes/Summary field
  followed by the target information. The target information is located in the
  SystemAttributes/SourceInfo/TargetInfo element. The target fields that
  will be included are TargetName, TargetType, and TargetURL.
- Severity will be set based on the contents of the
  SystemAttributes/SeverityCode field. If the field is set to FATAL, the
  severity will be set to 1. If the field is set to CRITICAL, the severity will be set
to 2. If the field is set to WARNING, the severity will be set to 3. For all other
  values the severity will be set to 5.
- Assignment Group will be hard coded to the Triage team.

The mappings for a normal update (not cleared) will be the same for both
templates as well. Listed below are the mappings that have been identified for the
normal update operation.

- Identifier will be set to the contents of the TicketID field
- Severity will be set based on the contents of the
  SystemAttributes/SeverityCode field. The mapping used here will be the
  same as the create operation.
- History Entry will be set based on the SystemAttributes/Severity field. The
  history entry will read "Updates from Enterprise Manager. The severity is now
  set to" followed by the SystemAttributes/Severity field.

The mappings for a close operation will be different depending on the template.
The template that is not closing the ticket, will use the same mapping as the
normal update. The mapping for the template that will close the ticket is listed
below.

- Identifier will be set to the contents of the TicketID field
- Status will be set to CLOSED
- History Entry will be set to "Corresponding incident in Enterprise Manager
  has been cleared."

4. Create the first template.

The template that does not close the ticket will be simpler to implement; so, we
will work with it first.

We start the process by copying the contents of the template skeleton from
Example 1–6, "Template Skeleton" into our editor. Then we develop the mapping
for the create and update requests as shown in Create Mappings and Update
Mappings.

Completed First Template Example shows what the template looks like after we
have added our mappings to the skeleton template. Also included in the section
are the results of our testing using the sample transactions in Appendix F, "Sample
Incident Data." It is important to test the first template thoroughly before using it
as a baseline to create other templates.

5. Create the second template.
Most of the work was done when we created the first template. To create the second template, we made a copy of the MyApp_Default_Incident.xsl file and renamed the copied file to MyApp_Default_Incident_AutoClose.xsl.

The only difference between the two templates is in the mapping for the close operation. The first field that needs to be mapped is the Identifier. Listed below is the mapping for the field.

<Identifier><xsl:value-of select="emcf:TicketID"/></Identifier>

The next field to be set is the Status field. This will be hard coded to CLOSED.

<Status>CLOSED</Status>

The last field to set is the HistoryEntry field. This will be hard coded to indicate that the incident in Enterprise Manager has been cleared.

<HistoryEntry>Corresponding incident in Enterprise Manager has been cleared.</HistoryEntry>

We added a check to determine whether the update is a close or a normal update. If the SeverityCode is set to CLEAR then we know we need to close the ticket, otherwise we will handle it as a normal update. Listed below is the check that was added.

<xsl:choose>
  <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'CLEAR'">
  
  Completed Second Template Example shows what the template looks like after we have added our mappings to handle the ticket close. Also included in the section are the results of our testing using the sample close transaction in Appendix F, "Sample Incident Data."

D.1 Create Mappings

The first mapping that we will develop is for create requests. We will identify the translation logic required for one field at a time. The first field is the Summary field. Listed below is the translation logic required to map this field to the Summary field in the incident.

<Summary><xsl:value-of select="emcf:SystemAttributes/emcf:Summary"/></Summary>

The next field to map is the Description field. It will be comprised of the Summary followed by the target information. Listed below is the mapping for the Description field.

<Description>Incident created in Enterprise Manager: <xsl:value-of select="emcf:SystemAttributes/emcf:Summary"/>

Target information:
  Target Type: <xsl:value-of select="emcf:SystemAttributes/emcf:SourceInfo/emcf:TargetInfo/emcf:TargetType"/>
  Target Name: <xsl:value-of select="emcf:SystemAttributes/emcf:SourceInfo/emcf:TargetInfo/emcf:TargetName"/>
  Target URL: <xsl:value-of select="emcf:SystemAttributes/emcf:SourceInfo/emcf:TargetInfo/emcf:TargetURL"/>
</Description>

The next field to map is the Severity field. Since it will be mapped for the create and update requests, it will be handled a little differently. A variable will be created to hold
the mapped severity value and the variable will be used in both mappings. This way there will only be one place to change if the mapping ever changes.

```xml
<xsl:variable name="sev">
  <xsl:choose>
    <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'FATAL'">1</xsl:when>
    <xsl:otherwise>5</xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<Severity><xsl:value-of select="$sev"/></Severity>
```

The last field to map is the AssignmentGroup field. This one is being hard coded since the incident does not have any information that can be used to determine the appropriate group in the external ticketing application.

```xml
<AssignmentGroup>Triage</AssignmentGroup>
```

D.2 Update Mappings

The first field to map is the Identifier field. It will be set to the incident TicketID field. Listed below is the mapping for this field.

```xml
<Identifier><xsl:value-of select="emcf:TicketID"/></Identifier>
```

The next field to map is the Severity field. Since the $sev variable was created earlier, it will just require setting the field to the variable contents.

```xml
<Severity><xsl:value-of select="$sev"/></Severity>
```

The last field is the HistoryEntry field. Notice that it is using Severity instead of SeverityCode. The SeverityCode field is an internal representation of the severity values and has fixed values. The Severity field in the incident data is locale specific so the value in this field will vary depending on the locale. The SeverityCode field should be used when checking for certain severity values and the Severity field should be used when displaying the severity.

```xml
<HistoryEntry>Updates from Enterprise Manager. The severity is now set to <xsl:value-of select="emcf:SystemAttributes/emcf:Severity"/></HistoryEntry>
```

D.3 Completed Templates

The following examples show a completed template:

- Completed First Template Example
- Completed Second Template Example

D.3.1 Completed First Template Example

Listed below is the completed template using the mappings shown earlier in this section. The filename for the template is MyApp_Default_Incident.xsl.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

D-4 Oracle Enterprise Manager Connectors Integration Guide
Listed below is the output from the template using the "Sample Create Transaction" data from Appendix F, "Sample Incident Data," as input.

<?xml version="1.0" encoding="UTF-8"?>
<createRequest xmlns="http://samplecompany.com"
xmlns:emcf="http://xmlns.oracle.com/sysman/connector">
<Summary>Memory Utilization is 70.518%, crossed warning (20) or critical (30) threshold.</Summary>
>Description>Incident created in Enterprise Manager: Memory Utilization is
</createRequest>
</xsl:template>
</xsl:stylesheet>
70.518%, crossed warning (20) or critical (30) threshold.
Target information:
   Target Type: host
   Target Name: target1.mycompany.com
   Target URL: https://mytarget1.mycompany.com:5416/em/redirect?pageType=TARGET_HOMEPAGE&amp;targetName=target1.mycompany.com&amp;targetType=host
</Description>
</createRequest>

Listed below is the output from the template using the "Sample Update Transaction" data from Appendix F, "Sample Incident Data," as input.

<?xml version="1.0" encoding="UTF-8"?>
<updateRequest xmlns="http://samplecompany.com"
xmlns:emcf="http://xml.mycompany.com/sysman/connector">
  <Identifier>CASD-000002 (cr:0000002)</Identifier>
  <Severity>3</Severity>
  <HistoryEntry>Updates from Enterprise Manager. The severity is now set to Warning</HistoryEntry>
</updateRequest>

D.3.2 Completed Second Template Example

Listed below is the completed second template. Notice the addition of the Closed section.

<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:emcf="http://xmlns.oracle.com/sysman/connector">
  <xsl:template match="emcf:EMIncident">
    <xsl:variable name="sev">
      <xsl:choose>
        <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'FATAL'">1</xsl:when>
        <xsl:otherwise>5</xsl:otherwise>
      </xsl:choose>
    </xsl:variable>

    <!-- CREATE Request -->
    <createRequest xmlns="http://samplecompany.com">
      <Summary><xsl:value-of
select="emcf:SystemAttributes/emcf:Summary"/></Summary>
      <Description><xsl:value-of
select="emcf:SystemAttributes/emcf:Summary"/>
Target information:
   Target Type: <xsl:value-of
select="emcf:SystemAttributes/emcf:TargetInfo/emcf:TargetType"/>
   Target Name: <xsl:value-of
select="emcf:SystemAttributes/emcf:TargetInfo/emcf:TargetName"/>
   Target URL: <xsl:value-of
</createRequest>

D-6 Oracle Enterprise Manager Connectors Integration Guide
Listed below is the output from the template using the "Sample Close Transaction" data from Appendix F, "Sample Incident Data," as input.

```xml
<?xml version='1.0' encoding='UTF-8'?>
  <Identifier>CASD-000002 (cr:000002)</Identifier>
  <Status>CLOSED</Status>
  <HistoryEntry>Corresponding incident in Enterprise Manager has been cleared.</HistoryEntry>
</updateRequest>
```
Create Event Template Example

This appendix contains an example that shows how to create a `createEvent` request template for an event connector. Listed below are the steps that would be taken to create the template:

1. **Study Data in External Application**
2. **Study Data in Enterprise Manager**
3. **Determine the Mapping for the Template**
4. **Create the Template**
5. **Create Mappings**

See [Completed Template Examples](#) for examples of a completed template.

### E.1 Study Data in External Application

The first step is to study the web service for the external application and determine what fields are available and what the data looks like in those fields.

Study of the web service WSDL shows that the root element for create requests is `createRequest`, and the namespace is `http://samplecompany.com`. The WSDL also shows that there are four required fields on a create. These fields are `summary`, `description`, `severity`, and `priority`. The severity field must be a numeric value from 1 to 5. Where 1 is the highest and 5 is the lowest. The priority field must be set to High, Medium, or Low. We have decided to populate just the required fields on the create request.

### E.2 Study Data in Enterprise Manager

Now that we understand the data required to create events, we need to understand what data is available in Enterprise Manager and where it is located. Study of the data showed that all of the data that we need is located in the `SystemAttributes` element. There is a `Summary` element that has summary information and there is a `SeverityCode` field that can be used as well. Also included in the `SystemAttributes` element is target information that we would like to see in the event description.

### E.3 Determine the Mapping for the Template

Listed below are the mappings that have been identified for the `createEvent` template:

- **Summary** will be set to the contents of the `SystemAttributes/Summary` field.
Create the Template

- **Description** will be set to the contents of the `SystemAttributes/Summary` field followed by the target information. The target information is located in the `SystemAttributes/SourceInfo/TargetInfo` element. The target fields that will be included are `TargetName`, `TargetType`, and `TargetURL`.

- **Severity** will be set based on the contents of the `SystemAttributes/SeverityCode` field. If the field is set to `FATAL`, the severity will be set to 1. If the field is set to `CRITICAL`, the severity will be set to 2. If the field is set to `WARNING`, the severity will be set to 3. For all other values the severity will be set to 5.

- **Priority** will be set based on the contents of the `SystemAttributes/SeverityCode` field. If the field is set to `FATAL` or `CRITICAL`, then the priority will be set to `High`. If the field is set to `WARNING`, then the priority will be set to `Medium`. All other values it will be set to `Low`.

### E.4 Create the Template

We start the process by copying the contents of the template skeleton from Example 2–4 into our editor. Then we develop the mapping for the create request as shown in Create Mappings. After the mapping has been done, save the file as `createEvent_request.xsl` and test the template using the sample transactions in Appendix G, “Sample Event Data,” to verify that it works.

Completed Template Examples shows what the template looks like after the mappings have been added to the skeleton template. Also included in the section are the results of our testing using the sample event transactions.

### E.5 Create Mappings

Listed below are the mapping for create requests. We will identify the translation logic required for one field at a time.

- **Mapping the Summary Field**

- **Mapping the Description Field**

- **Mapping the Severity Field**

- **Mapping the Priority Field**

#### E.5.1 Mapping the Summary Field

The first field is the Summary field. Listed below is the translation logic required to map this field to the Summary field in the event.

```
<Summary><xsl:value-of select="emcf:SystemAttributes/emcf:Summary"/></Summary>
```

#### E.5.2 Mapping the Description Field

The next field to map is the Description field. It will be comprised of the Summary followed by the target information. Listed below is the mapping for the Description field.

```
>Description><xsl:value-of select="emcf:SystemAttributes/emcf:Message"/>
Target information:
  Target Type: <xsl:value-of select="emcf:SystemAttributes/emcf:SourceInfo/emcf:TargetInfo/emcf:TargetType"/>
  Target Name: <xsl:value-of select="emcf:SystemAttributes/emcf:SourceInfo/emcf:TargetInfo/emcf:TargetName"/>
</Description>
```
E.5.3 Mapping the Severity Field

The next field to map is the Severity field. Listed below is the mapping for the Severity field.

```xml
<xsl:choose>
  <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'FATAL'"><Severity>1</Severity></xsl:when>
  <xsl:otherwise><Severity>5</Severity></xsl:otherwise>
</xsl:choose>
```

E.5.4 Mapping the Priority Field

The last field to map is the Priority field.

```xml
<xsl:choose>
  <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'FATAL'"><Priority>High</Priority></xsl:when>
  <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'CRITICAL'"><Priority>High</Priority></xsl:when>
  <xsl:when test="emcf:SystemAttributes/emcf:SeverityCode = 'WARNING'"><Priority>Medium</Priority></xsl:when>
  <xsl:otherwise><Priority>Low</Priority></xsl:otherwise>
</xsl:choose>
```

E.6 Completed Template Examples

Example E–1 shows the completed template using the mappings shown earlier in this section. The filename for the template is `createEvent_request.xsl`

Example E–1  Completed Event Template Sample

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:emcf="http://xmlns.oracle.com/sysman/connector">

  <xsl:template match="emcf:EMEvent">
    <createRequest xmlns="http://samplecompany.com">
      <!-- Set the summary to the EM event message -->
      <Summary><xsl:value-of select="emcf:SystemAttributes/emcf:Message"/></Summary>

      <!-- Set the description to the EM event message followed by the target information -->
      <Description>Event created in Enterprise Manager: <xsl:value-of select="emcf:SystemAttributes/emcf:Summary"/>
      Target information:
        Target Type: <xsl:value-of select="emcf:SystemAttributes/emcf:TargetInfo/emcf:TargetType"/>
        Target Name: <xsl:value-of select="emcf:SystemAttributes/emcf:TargetInfo/emcf:TargetName"/>
        Target URL: <xsl:value-of select="emcf:SystemAttributes/emcf:TargetInfo/emcf:TargetURL"/>
    </xsl:template>
</xsl:stylesheet>
```
Example E–2 shows the output from the template using the “Sample Create Transaction” data from Appendix G, "Sample Event Data," as input.

Example E–2 Event Template Output Sample

```xml
<?xml version="1.0" encoding="UTF-8"?>
<createRequest xmlns="http://samplecompany.com"
xmlns:emcf="http://xmlns.oracle.com/sysman/connector">
  <Summary>Memory Utilization is 69.913%, crossed warning (40) or critical (99) threshold.</Summary>
  <Description>Event created in Enterprise Manager: Memory Utilization is 69.913%, crossed warning (40) or critical (99) threshold.
Target information:
Target Type: host
Target Name: target.mycompany.com
Target URL: https://target.mycompany.com:5416/em/redirect?pageType=TARGET_HOMEPAGE&amp;targetName=target.mycompany.com&amp;targetType=host</Description>
  <Severity>3</Severity>
  <Priority>Medium</Priority>
</createRequest>
```
This appendix shows the following incident data samples:

- Sample Create Transaction
- Sample Update Transaction
- Sample Close Transaction

### F.1 Sample Create Transaction

```xml
<EMIncident xmlns="http://xmlns.oracle.com/sysman/connector">
  <ConnectorGUID>E99C9715E7C9485892F3F43F6CFAAB54</ConnectorGUID>
  <TicketID/>
  <HDUser>oracleoem</HDUser>
  <NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
  <NotificationRuleName>CASD Memory Util</NotificationRuleName>
  <ReopenTicket>No</ReopenTicket>
  <ConnectorVariable>
    <VariableName>TICKET_ID</VariableName>
    <VariableValue>TKT00001</VariableValue>
  </ConnectorVariable>
  <Property>
    <Name>AuthenticationType</Name>
    <Value>HTTPBasicAuthentication</Value>
  </Property>
  <Property>
    <Name>IsNewTargetType</Name>
    <Value>no</Value>
  </Property>
  <SystemAttributes>
    <IncidentID>F13444F23726231FE0431738F20AE9B0</IncidentID>
    <SourceInfo>
      <SourceObjInfo>
        <ObjID>A33ECDABD232AC3EAE2A71B81BB7E3A6</ObjID>
        <ObjName>slc00ykn.us.oracle.com</ObjName>
        <ObjOwner>SYSMAN</ObjOwner>
      </SourceObjInfo>
      <SourceObjType>Targets</SourceObjType>
      <SourceObjSubType>host</SourceObjSubType>
    </SourceObjInfo>
    <TargetInfo>
      <TargetGUID>A33ECDABD232AC3EAE2A71B81BB7E3A6</TargetGUID>
      <TargetName>slc00ykn.us.oracle.com</TargetName>
      <TargetType>host</TargetType>
      <TargetTypeLabel>Host</TargetTypeLabel>
    </TargetInfo>
  </SystemAttributes>
</EMIncident>
```
<TargetURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=TARGET_HOMEPAGE&amp;targetName=slc00ykn.us.oracle.com&amp;targetType=host</TargetURL>

<TargetProperty>
    <Name>Target_Host</Name>
    <Value>slc00ykn.us.oracle.com</Value>
</TargetProperty>

<TargetProperty>
    <Name>Operating_System</Name>
    <Value>Linux</Value>
</TargetProperty>

<TargetProperty>
    <Name>Platform</Name>
    <Value>x86_64</Value>
</TargetProperty>

<TargetProperty>
    <Name>Target_Version</Name>
    <Value>5.8.0.0.0</Value>
</TargetProperty>

<IncidentURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=sdk-core-event-console-detailIncident&amp;issueID=F13444F23726231FE0431738F20AE9B0</IncidentURL>

<AutoClose>true</AutoClose>

<Owner/>

<ResolutionState>New</ResolutionState>

<Severity>Critical</Severity>

<SeverityCode>CRITICAL</SeverityCode>

<Summary>Memory Utilization is 70.518%, crossed warning (20) or critical (30) threshold.</Summary>

<CreationDate>2014-01-30T09:28:27.000-08:00</CreationDate>

<LastUpdatedDate>2014-01-30T09:28:27.000-08:00</LastUpdatedDate>

<Category>Capacity</Category>

<HasEMEvent>true</HasEMEvent>

<EMEvent>
    <ConnectorGUID>E99C9715E7C9485892F3F6CFAAB54</ConnectorGUID>
    <NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
    <NotificationRuleName>CASD Memory Util</NotificationRuleName>
    <ConnectorVariable>
        <VariableName>TICKET_ID</VariableName>
        <VariableValue>TKT00001</VariableValue>
    </ConnectorVariable>
    <VariableName>TICKET_ID</VariableName>
    <VariableValue>TKT00001</VariableValue>
    <Property>
        <Name>AuthenticationType</Name>
        <Value>HTTPBasicAuthentication</Value>
    </Property>
    <Property>
        <Name>IsNewTargetType</Name>
        <Value>no</Value>
    </Property>
    <SystemAttributes>
        <EventClass>Metric Alert</EventClass>
        <EventID>F10F73C29D0C1B16E0431738F20AE9B0</EventID>
        <SequenceID>F10F73C29D0D1B16E0431738F20AE9B0</SequenceID>
        <ReportedDate>2014-01-30T09:28:26.000-08:00</ReportedDate>
<DisplayTZ>PST8PDT</DisplayTZ>
<EventName>Load:memUsedPct</EventName>
<Severity>Critical</Severity>
<SeverityCode>CRITICAL</SeverityCode>

<SourceInfo>
  <SourceObjInfo>
    <ObjID>A33ECDABD232AC3EAE2A71B83BB7E1A6</ObjID>
    <ObjName>slc00ykn.us.oracle.com</ObjName>
    <ObjOwner>SYSMAN</ObjOwner>
    <SourceObjType>TARGET</SourceObjType>
    <SourceObjSubType>host</SourceObjSubType>
  </SourceObjInfo>
  <TargetInfo>
    <TargetGUID>A33ECDABD232AC3EAE2A71B83BB7E1A6</TargetGUID>
    <TargetName>slc00ykn.us.oracle.com</TargetName>
    <TargetType>host</TargetType>
    <TargetTypeLabel>Host</TargetTypeLabel>
    <TargetURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=TARGET_HOMEPAGE&amp;targetName=slc00ykn.us.oracle.com&amp;targetType=host</TargetURL>
    <TargetProperty>
      <Name>Target_Host</Name>
      <Value>slc00ykn.us.oracle.com</Value>
    </TargetProperty>
    <TargetProperty>
      <Name>Operating System</Name>
      <Value>Linux</Value>
    </TargetProperty>
    <TargetProperty>
      <Name>Platform</Name>
      <Value>x86_64</Value>
    </TargetProperty>
    <TargetProperty>
      <Name>Target Version</Name>
      <Value>5.8.0.0.0</Value>
    </TargetProperty>
  </TargetInfo>
  <Message>Memory Utilization is 70.518%, crossed warning (20) or critical (30) threshold.</Message>
  <EventURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=sdk-core-event-console-detailEvent&amp;issueID=F10F73C29D0D1B16E0431738F20AEBAB</EventURL>
  <AutoClose>true</AutoClose>
  <EventCategory>Capacity</EventCategory>
</SourceInfo>

<EventClassSpecificAttributes>
  <StringAttribute>
    <Name>is_thresholdable</Name>
    <Value>1</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>coll_name</Name>
    <Value>LoadLinux</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>is_metric_extension</Name>
    <Value>0</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_column_resbundle</Name>
  </StringAttribute>
</EventClassSpecificAttributes>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description_resbundle</Name>
  <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
  <Name>unit_resbundle</Name>
  <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
  <Name>cyle_guid</Name>
  <Value>F10F73C29D091B16E0431738F20AEBAB</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_remote</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_type</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_guid</Name>
  <Value>86821B5F0CE858D6E4A7F7390E88B73C</Value>
</StringAttribute>
<StringAttribute>
  <Name>num_keys</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>key_value</Name>
  <Value> </Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description_nlsid</Name>
  <Value/>
</StringAttribute>
<StringAttribute>
  <Name>value</Name>
  <Value>70.518</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_long_running</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_group</Name>
  <Value>Load</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_udm</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description</Name>
  <Value/>
</StringAttribute>
<StringAttribute>
  <Name>metric_column_nlsid</Name>
  <Value/>
</StringAttribute>
F.2 Sample Update Transaction

<?xml version="1.0" encoding="UTF-8"?>
<EMIncident xmlns="http://xmlns.oracle.com/sysman/connector">
  <ConnectorGUID>E99C9715E7C9485892F3F43F6CFAAB54</ConnectorGUID>
  <TicketID>CASD-000002 (cr:0000002)</TicketID>
  <HDUser>oracleoem</HDUser>
  <NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
  <NotificationRuleName>CASD Memory Util</NotificationRuleName>
  <ReopenTicket>No</ReopenTicket>
  <ConnectorVariable>
    <VariableName>TICKET_ID</VariableName>
    <VariableValue>TKT00001</VariableValue>
  </ConnectorVariable>
  <Property>
    <Name>AuthenticationType</Name>
    <Value>HTTPBasicAuthentication</Value>
  </Property>
  <Property>
    <Name>InNewTargetType</Name>
    <Value>no</Value>
  </Property>
  <SystemAttributes>
    <IncidentID>F13444F23726231FE0431738F20AE9B0</IncidentID>
    <SourceInfo>
      <SourceObjInfo>
        <ObjID>A33ECDADB232AC3EAEE2A71B83BB7E3A6</ObjID>
        <ObjName>slc00ykn.us.oracle.com</ObjName>
        <ObjOwner>SYSMAN</ObjOwner>
        <SourceObjType>Targets</SourceObjType>
      </SourceObjInfo>
    </SourceInfo>
  </SystemAttributes>
</EMIncident>
<SourceObjSubType>host</SourceObjSubType>
</SourceObjInfo>
<TargetInfo>
<TargetGUID>A33ECDABD232AC3EAE2A71B83BB7E3A6</TargetGUID>
<TargetName>slc00ykn.us.oracle.com</TargetName>
<TargetType>host</TargetType>
<TargetTypeLabel>Host</TargetTypeLabel>
<TargetURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=TARGET_HOMEPAFILEtargetName=slc00ykn.us.oracle.com&amp;targetType=host</TargetURL>
<TargetProperty>
<Name>Target_Host</Name>
<Value>slc00ykn.us.oracle.com</Value>
</TargetProperty>
<TargetProperty>
<Name>Operating System</Name>
<Value>Linux</Value>
</TargetProperty>
<TargetProperty>
<Name>Platform</Name>
<Value>x86_64</Value>
</TargetProperty>
<TargetProperty>
<Name>Target Version</Name>
<Value>5.8.0.0.0</Value>
</TargetProperty>
</TargetInfo>
</SourceInfo>

<IncidentURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=sdk-core-event-console-detailIncident&amp;issueID=F13444F23726231FE0431738F20AE980</IncidentURL>
<AutoClose>true</AutoClose>
<Owner/>
<ResolutionState>New</ResolutionState>
<SeverityCode>WARNING</SeverityCode>
<UpdatedAttributes>sys_severity, sys_summary</UpdatedAttributes>
</SystemAttributes>
<HasEMEvent>true</HasEMEvent>
<EMEvent>
<ConnectorGUID>E99C9715E7C9485892F3F43F6CFAAB54</ConnectorGUID>
<NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
<NotificationRuleName>CASD Memory Util</NotificationRuleName>
<ConnectorVariable>
<VariableName>TICKET_ID</VariableName>
<VariableValue>TKT00001</VariableValue>
</ConnectorVariable>
<Property>
<Name>AuthenticationType</Name>
<Value>HTTPBasicAuthentication</Value>
</Property>
<Property>
<Name>...</Name>
<Value>...</Value>
</Property>

Sample Update Transaction
<Name>IsNewTargetType</Name>
<Value>no</Value>
</Property>

<EventAttributes>
<EventClass>Metric Alert</EventClass>
<EventID>F10F73DA20351B2E0431738F20AEB05</EventID>
<SequenceID>F10F73C29D01B16E0431738F20AEBAB</SequenceID>
<ReportedDate>2014-01-30T09:29:34.000-08:00</ReportedDate>
<DisplayTZ>PST8PDT</DisplayTZ>
<EventName>Load:memUsedPct</EventName>
<Severity>Warning</Severity>
<SeverityCode>WARNING</SeverityCode>

<Message>Memory Utilization is 70.824%, crossed warning (20) or critical (99) threshold.</Message>

<EventURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=TARGET_HOMEPAGE&amp;targetName=slc00ykn.us.oracle.com&amp;targetType=host</EventURL>

<AutoClose>true</AutoClose>
<EventCategory>Capacity</EventCategory>
</EventAttributes>
<Value>LoadLinux</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_metric_extension</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_column_resbundle</Name>
  <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description_resbundle</Name>
  <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description_nlsid</Name>
  <Value/>
</StringAttribute>
<StringAttribute>
  <Name>value</Name>
  <Value>70.824</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_long_running</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_group</Name>
  <Value>Load</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_udm</Name>
  <Value>
</StringAttribute>
F.3 Sample Close Transaction

<?xml version="1.0" encoding="UTF-8"?>
<EMIncident xmlns="http://xmlns.oracle.com/sysman/connector">
  <ConnectorGUID>E99C9715E7C9485892F3F43F6CFAAB54</ConnectorGUID>
  <TicketID>CASD-000002 (cr:0000002)</TicketID>
  <HDUser>oracleoem</HDUser>
  <NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
  <NotificationRuleName>CASD Memory Util</NotificationRuleName>
  <ReopenTicket>No</ReopenTicket>
  <ConnectorVariable>
    <VariableName>TICKET_ID</VariableName>
    <VariableValue>TKT00001</VariableValue>
  </ConnectorVariable>
  <Property>
    <Name>AuthenticationType</Name>
    <Value>HTTPBasicAuthentication</Value>
  </Property>
  <Property>
    <Name>InNewTargetType</Name>
    <Value>No</Value>
  </Property>
</EMIncident>
F-10  Oracle Enterprise Manager Connectors Integration Guide
Memory Utilization is 69.906%, fallen below warning (98) and critical (99) thresholds.
<EventClassSpecificAttributes>
  <StringAttribute>
    <Name>is_thresholdable</Name>
    <Value>1</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>coll_name</Name>
    <Value>LoadLinux</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>is_metric_extension</Name>
    <Value>0</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_column_resbundle</Name>
    <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_description_resbundle</Name>
    <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>unit_resbundle</Name>
    <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>cycle_guid</Name>
    <Value>F10F73C29D091B16E0431738F20AEBAB</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>is_remote</Name>
    <Value>0</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_type</Name>
    <Value>0</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_guid</Name>
    <Value>86821B5F0CE858D6E4A7F7390E88B73C</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>num_keys</Name>
    <Value>0</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>key_value</Name>
    <Value> </Value>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_description_nlsid</Name>
    <Value> </Value>
  </StringAttribute>
  <StringAttribute>
    <Name>value</Name>
    <Value>69.906</Value>
  </StringAttribute>
  <StringAttribute>
    <Name>is_long_running</Name>
    <Value>0</Value>
  </StringAttribute>
</EventClassSpecificAttributes>
</StringAttribute>
  </StringAttribute>
  <StringAttribute>
    <Name>metric_group</Name>
    <Value>Load</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>is_udm</Name>
    <Value>0</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>metric_description</Name>
    <Value/>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>metric_column_nlsid</Name>
    <Value>host_load_memUsedPct</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>metric_column</Name>
    <Value>Memory Utilization (%)</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>unit_nlsid</Name>
    <Value>em__sys__standard_percent</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>unit</Name>
    <Value>%</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>metric_group_nlsid</Name>
    <Value>host_load</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>metric_group_resbundle</Name>
    <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
  </StringAttribute>
</StringAttribute>
  <StringAttribute>
    <Name>severity_guid</Name>
    <Value>F10F73CC88D1B14E0431738F20ADBF0</Value>
  </StringAttribute>
</StringAttribute>
</EventClassSpecificAttributes>
</EMEvent>
</EMIncident>
This appendix shows the following event data transaction samples:

- Sample Create Transaction
- Sample Update Transaction
- Sample Close Transaction

G.1 Sample Create Transaction

```xml
<?xml version='1.0' encoding='UTF-8'?>
<EMEvent xmlns='http://xmlns.oracle.com/sysman/connector'>
  <ConnectorGUID>0091F8B08E8C41A8B87328EC6D6B3D50</ConnectorGUID>
  <NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
  <NotificationRuleName>OMU Memory Utilization %</NotificationRuleName>
  <Property>
    <Name>Notification_Method_Name</Name>
    <Value>OMU</Value>
  </Property>
  <Property>
    <Name>AuthenticationType</Name>
    <Value>HTTPBasicAuthentication</Value>
  </Property>
  <Property>
    <Name>IsNewTargetType</Name>
    <Value>yes</Value>
  </Property>
  <SystemAttributes>
    <EventClass>Metric Alert</EventClass>
    <EventID>F10F73C29D031B16E0431738F20AEBAB</EventID>
    <SequenceID>F10F73CCCC821B14E0431738F20ADBF0</SequenceID>
    <ReportedDate>2014-01-30T08:03:22.000-08:00</ReportedDate>
    <DisplayTZ>PST8PDT</DisplayTZ>
    <EventName>Load:memUsedPct</EventName>
    <Severity>Warning</Severity>
    <SeverityCode>WARNING</SeverityCode>
    <SourceInfo>
      <SourceObjInfo>
        <ObjID>A33ECDABD232AC3EAE2A71B83BB7E3A6</ObjID>
        <ObjName>slc00ykn.us.oracle.com</ObjName>
        <ObjOwner>SYSMAN</ObjOwner>
        <SourceObjType>TARGET</SourceObjType>
        <SourceObjSubType>host</SourceObjSubType>
      </SourceObjInfo>
      <TargetInfo>
        <TargetGUID>A33ECDABD232AC3EAE2A71B83BB7E3A6</TargetGUID>
      </TargetInfo>
    </SourceInfo>
  </SystemAttributes>
</EMEvent>
```
<TargetName>slc00yknu.oracle.com</TargetName>
<Name>Target_Host</Name>
<Value>slc00yknu.oracle.com</Value>
<Name>Operating System</Name>
<Value>Linux</Value>
<Name>Platform</Name>
<Value>x86_64</Value>
<Name>Target Version</Name>
<Value>5.8.0.0.0</Value>

<Message>Memory Utilization is 69.913%, crossed warning (40) or critical (99) threshold.</Message>
<EventURL>https://slc00yknu.oracle.com:5416/em/redirect?pageType=dk-core-event-console-detailEvent&amp;issueID=F10F73CCC821B14E0431738F20ADBF0</EventURL>
<EventCategory>Capacity</EventCategory>

<StringAttribute>
<Name>is_thresholdable</Name>
<Value>1</Value>
</StringAttribute>
<StringAttribute>
<Name>coll_name</Name>
<Value>LoadLinux</Value>
</StringAttribute>
<StringAttribute>
<Name>is_metric_extension</Name>
<Value>0</Value>
</StringAttribute>
<StringAttribute>
<Name>metric_column_resbundle</Name>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
<Name>metric_description_resbundle</Name>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
<Name>unit_resbundle</Name>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
<Name>cyle_guid</Name>
<Value>F10F73CCC87E1B14E0431738F20ADBF0</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_remote</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_type</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_guid</Name>
  <Value>86821b5f0cee858d6e4a7f7390e88b73c</Value>
</StringAttribute>
<StringAttribute>
  <Name>num_keys</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>key_value</Name>
  <Value></Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description_nlsid</Name>
  <Value></Value>
</StringAttribute>
<StringAttribute>
  <Name>value</Name>
  <Value>69.913</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_long_running</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_group</Name>
  <Value>Load</Value>
</StringAttribute>
<StringAttribute>
  <Name>is_udm</Name>
  <Value>0</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_description</Name>
  <Value></Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_column_nlsid</Name>
  <Value>host_load_memUsedPct</Value>
</StringAttribute>
<StringAttribute>
  <Name>metric_column</Name>
  <Value>Memory Utilization (%)</Value>
</StringAttribute>
<StringAttribute>
  <Name>unit_nlsid</Name>
  <Value>em__sys__standard_percent</Value>
</StringAttribute>
<StringAttribute>
  <Name>unit</Name>
  <Value>%</Value>
</StringAttribute>
G.2 Sample Update Transaction

<?xml version="1.0" encoding="UTF-8"?>
<EMEvent xmlns="http://xmlns.oracle.com/sysman/connector">
  <ConnectorGUID>0091F8B08B8C41A88B7128EC6D6B3D50</ConnectorGUID>
  <ExternalEventID>MOCK-ID-001</ExternalEventID>
  <NotificationRuleOwner>SYSMAN</NotificationRuleOwner>
  <NotificationRuleName>OMU Memory Utilization %</NotificationRuleName>
  <Property>
    <Name>Notification_Method_Name</Name>
    <Value>OMU</Value>
  </Property>
  <Property>
    <Name>AuthenticationType</Name>
    <Value>HTTPBasicAuthentication</Value>
  </Property>
  <Property>
    <Name>IsNewTargetType</Name>
    <Value>yes</Value>
  </Property>
  <SystemAttributes>
    <EventClass>Metric Alert</EventClass>
    <EventID>F10F73DA202A1B12E0431738F20AEB05</EventID>
    <SequenceID>F10F73CCC8821B14E0431738F20AEBF0</SequenceID>
    <ReportedDate>2014-01-30T08:04:43.000-08:00</ReportedDate>
    <DisplayTZ>PST8PDT</DisplayTZ>
    <EventName>Load:memUsedPct</EventName>
    <Severity>Critical</Severity>
    <SeverityCode>CRITICAL</SeverityCode>
  </SystemAttributes>
</EMEvent>
<Name>Target_Host</Name>
<Value>slc00ykn.us.oracle.com</Value>
</TargetProperty>
<TargetProperty>
<Name>Operating System</Name>
<Value>Linux</Value>
</TargetProperty>
<TargetProperty>
<Name>Platform</Name>
<Value>x86_64</Value>
</TargetProperty>
<TargetProperty>
<Name>Target Version</Name>
<Value>5.8.0.0.0</Value>
</TargetProperty>
</TargetInfo>
</SourceInfo>
<Message>Memory Utilization is 69.834%, crossed warning (40) or critical (50) threshold.</Message>

<EventURL>https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=sdk-core-event-console-detailEvent&amp;issueID=F10F73CCC87E1B14E0431738F20ADBF0</EventURL>

<AutoClose>true</AutoClose>
<EventCategory>Capacity</EventCategory>
</SystemAttributes>
<EventClassSpecificAttributes>
<StringAttribute>
<Name>is_thresholdable</Name>
<Value>1</Value>
</StringAttribute>
<StringAttribute>
<Name>coll_name</Name>
<Value>LoadLinux</Value>
</StringAttribute>
<StringAttribute>
<Name>is_metric_extension</Name>
<Value>0</Value>
</StringAttribute>
<StringAttribute>
<Name>metric_column_resbundle</Name>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
<Name>metric_description_resbundle</Name>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
<Name>unit_resbundle</Name>
<Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>
<StringAttribute>
<Name>cycle_guid</Name>
<Value>F10F73CCC87E1B14E0431738F20ADBF0</Value>
</StringAttribute>
<StringAttribute>
<Name>is_remote</Name>
<Value>0</Value>
</StringAttribute>
<StringAttribute>
<Name>metric_type</Name>
G.3 Sample Close Transaction

<?xml version="1.0" encoding="UTF-8"?>
<EMEvent xmlns="http://xmlns.oracle.com/sysman/connector">
</EMEvent>
Memory Utilization is 69.518%, fallen below warning (98) and critical (99) thresholds.

EventURL=https://slc00ykn.us.oracle.com:5416/em/redirect?pageType=sdk-core-event-console-detailEvent&amp;issueID=F10F73CCC8821B14E0431738F20ADBFO
<StringAttribute>
  <Name>num_keys</Name>
  <Value>0</Value>
</StringAttribute>

<StringAttribute>
  <Name>key_value</Name>
  <Value> </Value>
</StringAttribute>

<StringAttribute>
  <Name>metric_description_nlsid</Name>
  <Value>/</Value>
</StringAttribute>

<StringAttribute>
  <Name>is_long_running</Name>
  <Value>0</Value>
</StringAttribute>

<StringAttribute>
  <Name>metric_group</Name>
  <Value>Load</Value>
</StringAttribute>

<StringAttribute>
  <Name>is_udm</Name>
  <Value>0</Value>
</StringAttribute>

<StringAttribute>
  <Name>metric_description</Name>
  <Value>/</Value>
</StringAttribute>

<StringAttribute>
  <Name>metric_column_nlsid</Name>
  <Value>host_load_memUsedPct</Value>
</StringAttribute>

<StringAttribute>
  <Name>unit_nlsid</Name>
  <Value>em__sys__standard_percent</Value>
</StringAttribute>

<StringAttribute>
  <Name>unit</Name>
  <Value>%</Value>
</StringAttribute>

<StringAttribute>
  <Name>metric_group_nlsid</Name>
  <Value>host_load</Value>
</StringAttribute>

<StringAttribute>
  <Name>metric_group_resbundle</Name>
  <Value>oracle.sysman.eml.rsc.gen.hostMsg</Value>
</StringAttribute>

<StringAttribute>
  <Name>severity_guid</Name>
  <Value>F10F73CCC8841B14E0431738F20ADBF0</Value>
</StringAttribute>
Sample Close Transaction

</EventClassSpecificAttributes>
</EMEvent>
This glossary defines the different event types.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>StringT64</td>
<td>The name of the connector type</td>
</tr>
<tr>
<td>Version</td>
<td>VersionT</td>
<td>The version of the connector type</td>
</tr>
<tr>
<td>EMCompatibleVersion</td>
<td>VersionT</td>
<td>The EM compatibility version of the connector type</td>
</tr>
<tr>
<td>Description</td>
<td>StringT256</td>
<td>The description of the connector type</td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td>The category of the connector type. It must be one of the three values listed next.</td>
</tr>
<tr>
<td>NewTargetType</td>
<td></td>
<td>New target type definition for event connectors. This target type will be registered with Enterprise Manager and target instances can be created subsequently, including a default target. These targets are used to accommodate external events.</td>
</tr>
<tr>
<td>TargetTypeName</td>
<td>StringStrictT64</td>
<td>The name of the target type</td>
</tr>
<tr>
<td>targetTypeDisplayName</td>
<td>StringT128</td>
<td>The name of the target type, as shown on UI</td>
</tr>
<tr>
<td>DefaultTargetName</td>
<td>StringStrictT256</td>
<td>The name of the default target of the target type. The default target will be used as a generic bucket to hold external events.</td>
</tr>
<tr>
<td>DefaultTargetDisplayName</td>
<td>StringT256</td>
<td>The name of the default target of the target type, to be displayed on UI.</td>
</tr>
<tr>
<td>SOAPHeaderAuthentication</td>
<td>SOAPHeaderAuthenticationType</td>
<td>Specification for SOAP Header authentication.</td>
</tr>
<tr>
<td>HTTPBasicAuthentication</td>
<td>UsernamePasswordAuthenticationType</td>
<td>Specification for HTTP basic authentication.</td>
</tr>
<tr>
<td>UserNameTokenAuthentication</td>
<td>UsernamePasswordAuthenticationType</td>
<td>Specification for Username Token authentication.</td>
</tr>
<tr>
<td>ConfigVariable</td>
<td>ConfigVariableType</td>
<td>The variables used during connector configuration. These variables are required by external system to complete connector configuration, which includes registering with the external system. For instance, one configuration variable can be the resolution state required by Microsoft Operation Manager.</td>
</tr>
<tr>
<td>ConnectivityTestVariable</td>
<td>ConfigVariableType</td>
<td>An optional variable used to test connection to an external server.</td>
</tr>
<tr>
<td>Service</td>
<td>ServiceType</td>
<td>Specification for web services, which define how connector framework can communicate with external system.</td>
</tr>
<tr>
<td>ExternalURL</td>
<td>ExternalURLType</td>
<td>Specification for the URL link to the external server, including the URL pattern and server specific variables. It is used to provide links to external server for viewing ticket details.</td>
</tr>
</tbody>
</table>
TemplateRegistration

TemplateRegistrationType

Specific ation for template registration. A template is registered based on the information provided in the element. A connector deployment descriptor can have an optional list of up to 50 template registration elements.

Method

- The name of the web service method. Each connector category has a predefined set of methods as defined next.

WebServiceEndpoint

StringT256

The web service end point indicating a specific location for accessing a service.

SOAPAction

StringT64

The SOAP action which carries out the web service call for the method.

SOAPBindingType

- The type of SOAP over HTTP binding. Choose from one of the four options defined next

Username

ConfigVariableType

The username of the authentication.

Password

ConfigVariableType

The password of the authentication

AuthVariable

ConfigVariableType

An optional list of extra authentication variables besides username and password

SOAPHeader

StringT256

A SOAP header string serving as template for the SOAP header. It is to be updated with user inputs for variables defined above and bound with a HTTP request.

VariableName

StringStrictT32

Name of the variable

DisplayName

StringT64

Name of the variable used for display on UI.

Pattern

StringT256

The URL pattern used to format links to the external server.

FileName

StringT256

The template file name

InternalName

StringStrictT128

A name representing the template in the connector framework

TemplateName

StringStrictT128

The template display name to be used on UI.

TemplateType

- The template type as one of the three options defined next.

Description

StringT512

A description of the template
Index

A
Add Target Page, data exchange connectors, 3-7
architecture of data exchange connectors, 3-1
auto ticketing, 1-1

C
clearing a ticket, 1-3
complex response properties, 4-13
connector
  event, 2-1
creating a ticket, 1-2

D
data exchange
  OBAM artifacts for inbound session, 3-40
data exchange connectors, 3-38
    Add Target Page, 3-7
    architecture, 3-1
    checking logs, 3-45
    creating a data exchange hub, 3-6
    data exchange hub, 3-3
    data flow tips, 3-44
denormalized message format, 3-4
EM-BAM data objects, 3-34
EM-BAM EMS definitions, 3-35
importing OBAM artifacts, 3-34
inbound alert schema, 3-33
inbound indicators schema, 3-31
inbound JMS topics, 3-31
inbound message schemas, 3-31
  JNDI details, 3-44
message example defaults, 3-32
normalized message format, 3-4
notification method, 3-44
notification rules, 3-44
qualified XML message sample, 3-31
Select Business Events/Indicators Page, 3-28
Session Setup Page, 3-7, 3-28
setting up, 3-5
setting up data flow, 3-34
unqualified XML message sample, 3-31
updating JNDI, 3-38
debugging, B-4
debug option, B-4
view debug messages, B-4
denormalized message format, outbound message schema, 3-22
deploy
  event connector, 2-17
ticketing connector, 1-21

E
EM-BAM
data objects, 3-34, 3-38
EMS definitions, 3-35
enterprise link plans, 3-38
EM-BAM data objects, 3-38
EM-BAM enterprise link plans, 3-38
error messages, B-1
event connector, 2-1
building, 2-4
define connector descriptor file, 2-9
determine connector functionality, 2-4
developing required template files, 2-5
how event connector functions, 2-2
metadata file categories, 2-1
package and deploy, 2-17
prerequisites, 2-3
recommended template filenames, 2-5
extracting schema files, 1-4, 2-3

I
importing OBAM artifacts, 3-34
inbound alert schema, 3-33
inbound indicators schema, 3-31

J
Java Messag Service (JMS)
suggested reading, 3-46
JNDI details, data exchange connectors, 3-44

M
manual ticketing, 1-1
message example defaults, 3-32
Metadata File Categories, 1-2
N
normalized alert message, outbound message schema, 3-17
normalized message format, outbound message schema, 3-13
normalized metric data message, outbound message schema, 3-16
normalized metric message, outbound message schema, 3-14
normalized security filter message, outbound message schema, 3-15
normalized target message, outbound message schema, 3-13
notification method, data exchange connectors, 3-44
notification rules
  data exchange connectors, 3-44

O
OBAM
  artifacts for inbound session, 3-40
  EM-BAM data objects, 3-34, 3-38
  EM-BAM EMS definitions, 3-35
  EM-BAM enterprise links plans, 3-38
  importing artifacts, 3-34
  setting up data flow, 3-34
  updating JNDI, 3-38
outbound message schema
denormalized message format, 3-22
normalized alert message, 3-17
normalized message format, 3-13
normalized metric data message, 3-16
normalized metric message, 3-14
normalized security filter message, 3-15
normalized target message, 3-13

P
package
  event connector, 2-17
  ticketing connector, 1-21
prerequisites
  event connector, 2-3
  ticketing connector, 1-4

Q
qualified XML message sample, 3-31

R
reference tables, 4-1
response file properties
  Windows, 4-8

S
Select Business Events/Indicators Page, data exchange connectors, 3-28
Session Setup Page, data exchange connector, 3-7

status codes, 4-14
  job status codes, 4-14
Synchronizing Ticket Status, 1-3

ticketing connector, 1-1
  auto ticketing, 1-1
  building, 1-4
  clearing a ticket, 1-3
  configuring, 1-2
  connector descriptor file, 1-2
  creating a ticket, 1-2
  default templates, 1-8
  defining connector descriptor file, 1-12
  determining functionality, 1-4
  developing required template files, 1-6
  manual ticketing, 1-1
  package and deploy, 1-21
  prerequisites, 1-4
  updating a ticket, 1-3

unqualified XML message sample, 3-31
  updating a ticket, 1-3
  updating JNDI, 3-38