Oracle® Enterprise Manager

Connectors Integration Guide

11*g* Release 1 (11.1.0.0)

E17065-02

February 2011

ORACLE®

Oracle Enterprise Manager Connectors Integration Guide, 11*g* Release 1 (11.1.0.0)

E17065-02

Primary Author: Leo Cloutier

Contents

3 Building a Data Exchange Connector

Preface

This Preface contains these sections:

Audience

The Oracle Enterprise Manager Integration Guide is intended for system integrators who want to integrate other management systems with Enterprise Manager.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711

Related Documentation

For information on setting up the environment for RAC provisioning jobs, see the following manuals in the Oracle Database 10*g* Release 2 Documentation Library:

- *Oracle Clusterware and Oracle Real Application Clusters Installation Guide for Linux*
- *Oracle Clusterware and Oracle Real Application Clusters Installation and Configuration Guide for Microsoft Windows Platforms*
- *Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide*

For information on Enterprise Manager, see the following manuals in the Oracle Enterprise Manager 10*g* Release 2 Documenation Library:

- *Oracle Database 2 Day DBA*
- *Oracle Enterprise Manager Concepts*
- *Oracle Enterprise Manager Quick Installation Guide*
- *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*
- *Oracle Enterprise Manager Advanced Configuration*
- *Oracle Enterprise Manager Configuration for Oracle Collaboration Suite*
- *Oracle Enterprise Manager SNMP Support Reference Guide*
- *Oracle Enterprise Manager Policy Reference Manual*
- *Oracle Enterprise Manager Metric Reference Manual*
- *Oracle Enterprise Manager Command Line Interface*
- *Extending Oracle Enterprise Manager*

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

`http://www.oracle.com/technology/documentation/oem.html`

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

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation/`

Conventions

The following text conventions are used in this document:

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

1

Building a Help Desk Connector

This chapter provides information you need to build a help desk connector and integrate it with Enterprise Manager. This chapter has following sections:

Introduction

Enterprise Manager Release 11*g* provides a Management Connector Framework (referred to as Connector Framework) to allow developers to build help desk connectors based on metadata (XMLs and XSLs).

Help desk connectors created through the framework inherit the following features:

- **Auto Ticketing —** Lets you describe the connector to automatically open or update a ticket whenever an alert is triggered in Enterprise Manager. You can specify the set of alerts for which tickets must be opened and the alert severity for which this should happen. You can do this with Notification Rules, the user-defined rules that define the criteria by which notifications should be sent for alerts.
- **Manual Ticketing —** Lets you manually open a ticket from Enterprise Manager console based on an open alert in Enterprise Manager. The connector populates the ticket with details based on the alert and the ticket template.

To utilize these ticketing features for your own help desk system, you need to provide a set of metdata files. The categories of metadata files listed in Table 1–1 are required for building a help desk connector:

Table 1–1 Metadata File Categories

Category	Type	Description
Connector Descriptor	XML	Lets you customize the integration for your ticketing system. Through XML you describe how the configuration UI pages should be generated in this file.

Table 1–1 (Cont.) Metadata File Categories

Category	Type	Description
Ticket Templates	XSLT	Describe how a ticket is filled out in the context of an event. Ticket templates explain the mappings from Enterprise Manager alert data fields to the corresponding ticket data fields in ticket creation and update through the XSLT language.
Response Transformations	XSLT	Normalize the ticketing system Web service's response data into a format that the Ticketing Connector Framework can understand.

Prerequisites

Before building a help desk connector, ensure that your ticketing system meets the following pre-requisites:

- The ticketing system exposes the following Web services:
  - Create Ticket
  - Get Ticket
  - Update Ticket
- The Web service is of Document Literal style
- Web service supports Simple Object Access Protocol (SOAP) header-based authentication.

Defining XML and XSL Files

The section provides details of various XML and XSL files that you must define for building a help desk connector.

Oracle recommends you to refer to the example metadata files available at the following location:

```
$OMS_HOME/sysman/connector/Remedy_Connector
```

Note: Do not make changes to any of the files in this location.

Connector Descriptor XML File

Define a connector descriptor XML file to describe the connector metadata and the configuration properties of the connector such as Web service end points, authentication schema, and ticket URL pattern. This XML file defines how the configuration pages are dynamically generated for the connector.

Example 1–1 conforms to the connectorType.XSD file. You can use it as a template by replacing the highlighted elements with proper values.

Note: This sample file is located in the following directory:

```
$ORACLE_HOME/sysman/connector/common/schema/
connectorType.xsd
```

The descriptor XML file for Remedy Connector is named `RemedyDeploy.xml`.

Example 1–1 Sample Connector Descriptor XML File

```
<?xml version='1.0' encoding='UTF-8'?>
<ManagementConnector xmlns="http://xmlns.oracle.com/sysman/connector">
  <Name>specify the connector name</Name>
  <Version>specify the 5-digit version, e.g., 1.0.0.0.0</Version>
  <Description>specify a short description</Description>
  <!-- Category is fixed to TicketingConnector. -->
  <Category>TicketingConnector</Category>
  <TicketingConnector>
    <!-- authentication element is optional. If authentication is required, the
SOAP header should match the Data element in this section. Other elements are
displayed on the default UI to collect data to generate the Data element. -->
    <Authentication>
      <Username>
        <AttributeName>specify the username_element_name</AttributeName>
        <DisplayName>specify the label</DisplayName>
      </Username>
      <Password>
        <AttributeName>specify the password_element_name</AttributeName>
        <DisplayName>specify the display label</DisplayName>
      </Password>
      <Attribute>
        <AttributeName>specify attribute other than username and
password</AttributeName>
        <DisplayName>specify display label</DisplayName>
      </Attribute>
      <!-- Attribute element can repeat. -->
      <Data>
        <![CDATA[
          specify how authentication element should appear in SOAP header
        ]]>
      </Data>
    </Authentication>
    <TicketingService>
      <Method>createTicket</Method>
      <WebServiceEndpoint>
        <![CDATA[specify URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </TicketingService>
    <TicketingService>
      <Method>updateTicket</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </TicketingService>
  <!-- BaseURL element is for the default UI for ticketing connectors, the
following value should not be changed. -->
  <BaseURL>/em/console/connector/ticket/ticketConnectorConfig</BaseURL>
    <!-- ExternalURL is used to generate the ticket URL. -->
    <ExternalURL>
      <!-- Strings in brackets ([]) are user variables, i.e., the values to be
collected from UI. Strings in {} are substitution variables, i.e., the values to
be collected from the web service response (they need to have the same element
names as the ones from the response XML. The following is an example. -->
```

```
      <Pattern>
        <![CDATA[specify the URL as http://[Web
Server]/arsys/servlet/ViewFormServlet?form=[HelpDesk Case Form
Name]&server=[ARServer Name]&eid={Case_ID}]]>
      </Pattern>
      <UserVariable>specify the Web server</UserVariable>
      <UserVariable>specify the HelpDesk Case Form Name</UserVariable>
      <UserVariable>specify the ARServer Name</UserVariable>
      <SubstitutionVariable>specify the Case_ID</SubstitutionVariable>
    </ExternalURL>
  </TicketingConnector>
</ManagementConnector>
```

The Table 1–2 provides a break-down of the file and explains what each section does:

Table 1–2 Connector Descriptor XML File Explained

Metadata File Sections	Explanation
<Authentication> <Username> <AttributeName>specify the username_element_name</AttributeName> <DisplayName>specify the label</DisplayName> </Username> <Password> <AttributeName>specify the password_element_name</AttributeName> <DisplayName>specify the display label</DisplayName> </Password> <Attribute> <AttributeName>specify attribute other than username and password</AttributeName> <DisplayName>specify display label</DisplayName> </Attribute> <!-- Attribute element can repeat. --> <Data> <![CDATA[specify how authentication element should appear in SOAP header]]> </Data> </Authentication>	(Optional) This section allows you to describe the authentication schema of ticketing system Web services. The authentication section of the Configure Management Connector page (Figure 1–2) is generated based on this section. The username, password, and additional text fields are populated based on this section. ■ The <AttributeName> tag marks the name of the attribute in the Web services XML document. ■ The <DisplayName> tag describes how the corresponding text-box should be labeled.

Table 1–2 (Cont.) Connector Descriptor XML File Explained

Metadata File Sections	Explanation
<TicketingService> <Method>createTicket</Method> <WebServiceEndpoint> <![CDATA[***specify URL***]]> </WebServiceEndpoint> <SOAPAction>***specify the SOAP action***</SOAPAction> <Namespace>***specify the namespace***</Namespace> <NamespacePrefix>***specify the namespace prefix***</NamespacePrefix> </TicketingService> <TicketingService> <Method>updateTicket</Method> <WebServiceEndpoint> <![CDATA[***specify the URL***]> </WebServiceEndpoint> <SOAPAction>***specify the SOAP action***</SOAPAction> <Namespace>***specify the namespace***</Namespace> <NamespacePrefix>***specify the namespace prefix***</NamespacePrefix> </TicketingService> <BaseURL>/em/console/connector/ticket/ticketConnectorConfig</BaseURL>	This section allows you to specify configurations specific to the Ticketing System's Web services. For your Ticketing System's `create ticket` and `update ticket` Web services, you need to describe the corresponding SOAP action and namespace prefix in order to enable the framework to properly communicate with the Ticketing System. ■ The <WebServiceEndpoint> tag describes the default Web service endpoint string to be displayed in the Web service section of the Management Connector page (Figure 1–2). ■ <BaseURL> tag is for the default UI for help desk connectors. You should not edit the values.
<ExternalURL> <Pattern> <![CDATA[***specify the URL as http://[Web Server]/arsys/servlet/ViewFormServlet?form=[HelpDesk Case Form Name]&server=[ARServer Name]&eid={Case_ID}***]]> </Pattern> <UserVariable>***specify the Web server***</UserVariable> <UserVariable>specify the ***HelpDesk Case Form Name***</UserVariable> <UserVariable>***specify the ARServer Name***</UserVariable> <SubstitutionVariable>***specify the Case_ID***</SubstitutionVariable> </ExternalURL> </TicketingConnector>	This section enables you to configure the Ticket Connector Framework to generate a ticket URL to your unique Ticketing System. ■ The value of the <Pattern> tag describes the URL string, and how user configured variables are inserted into it. ■ A textbox label pair is inserted into the Webconsole section for each <UserVariable> tag. The values that the users provide for each user variable is inserted into the URL pattern string accordingly. If there is a user variable `"X"` then the user input value replaces `"[X]"` when the ticket URL is generated. ■ The <SubstitutionVariable> tag describes which tag from the Web service create ticket response should map into the URL pattern string. For example, if the Ticketing System create ticket Web service returns a tag <TicketID>, it will replace the text in the ticket URL pattern {TicketID}.

connectorType.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: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="xsd:string"/>
        <xsd:element name="Version" type="xsd:string"/>
        <xsd:element name="Description" type="xsd:string"/>
        <xsd:element name="Category">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="EventConnector"/>
              <xsd:enumeration value="TicketingConnector"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:choice>
          <xsd:element name="EventConnector" type="EventConnector"/>
          <xsd:element name="TicketingConnector" type="TicketingConnector"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="EventConnector">
    <xsd:sequence>
    <xsd:element name="IsNewTargetType" type="xsd:boolean"/>
      <xsd:element name="Authentication" type="Authentication" minOccurs="0"/>
      <xsd:element name="EventService" type="EventService" minOccurs="1"
maxOccurs="unbounded"/>
      <xsd:element name="TargetResponse" type="Response" minOccurs="0"/>
      <xsd:element name="BaseURL" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TicketingConnector">
    <xsd:sequence>
      <xsd:element name="Authentication" type="Authentication" minOccurs="0"/>
      <xsd:element name="TicketingService" type="TicketingService" minOccurs="1"
maxOccurs="unbounded"/>
      <xsd:element name="BaseURL" type="xsd:string"/>
      <xsd:element name="ExternalURL" type="ExternalURL"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="EventService">
    <xsd:sequence>
      <xsd:element name="Method" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="getNewAlerts"/>
            <xsd:enumeration value="getUpdatedAlerts"/>
            <xsd:enumeration value="acknowledgeAlerts"/>
            <xsd:enumeration value="setup"/>
            <xsd:enumeration value="initialize"/>
            <xsd:enumeration value="uninitialize"/>
            <xsd:enumeration value="cleanup"/>
            <xsd:enumeration value="updateAlerts"/>
          </xsd:restriction>
```

```
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="WebServiceEndpoint" type="xsd:string"/>
      <xsd:element name="SOAPAction" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Namespace" type="xsd:string"/>
      <xsd:element name="NamespacePrefix" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
 <xsd:complexType name="TicketingService">
    <xsd:sequence>
      <xsd:element name="Method" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="createTicket"/>
            <xsd:enumeration value="updateTicket"/>
            <xsd:enumeration value="getTicket"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="WebServiceEndpoint" type="xsd:string"/>
      <xsd:element name="SOAPAction" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Namespace" type="xsd:string"/>
      <xsd:element name="NamespacePrefix" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Authentication">
    <xsd:sequence>
      <xsd:element name="Username" type="Username"/>
      <xsd:element name="Password" type="Password"/>
      <xsd:element name="Attribute" type="Attribute" maxOccurs="unbounded"
minOccurs="0"/>
      <xsd:element name="Data" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Username">
    <xsd:sequence>
      <xsd:element name="AttributeName" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Password">
    <xsd:sequence>
      <xsd:element name="AttributeName" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Attribute">
    <xsd:sequence>
      <xsd:element name="AttributeName" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Response">
    <xsd:sequence>
      <xsd:element name="ScriptName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ExternalURL">
    <xsd:sequence>
      <xsd:element name="Pattern" type="xsd:string"/>
```

```
      <xsd:element name="UserVariable" type="xsd:string" maxOccurs="unbounded"
minOccurs="0"/>
      <xsd:element name="SubstitutionVariable" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Default Ticket Template XSL Files

Ticket templates describe how Enterprise Manager alerts should be transformed into the messages (SOAP body XML) expected by the ticketing system. To ensure that the tickets generated are compatible with your help desk, you can use ticket templates to specify how ticket fields should be pre-filled when opening or updating a ticket based on an Enterprise Manager alert.

You can add new ticket templates or modify existing ticket templates. You should provide at least one ticket template with your help desk connector. The template should include the logic on how to transform the following requests:

- Create Ticket
- Get Ticket
- Update Ticket
- Reopen Ticket
- Close Ticket

You can ship as many templates as needed. For example, you might choose to ship templates based on categories such as production database, development database, and test database, or priority, or combination of both. Oracle recommends that you ship at least two sets of templates: those that auto close and those that do not.

You can modify these templates or add more templates if required. You can delete the ticket templates from the user interface or upload them using `emctl`.

The fields available to map to a ticket are shown in Table 1–3:

Table 1–3 Enterprise Manager Data Fields

Data Fields	Description
EMUser	▪ For auto-ticketing, this is the notification rule owner. ▪ For manual ticketing, this is the console user that triggered the ticket creation.
HDUser	Helpdesk user registered with the Connector; this is same as the user name specified for the WS authentication.
TicketID	Identifies the ticket associated with the current alert (this is available after ticket creation).
ConnectorID	Identifies the help desk connector that processed the event and issued the ticket creation or ticket update.
TargetType	Type of target that the alert is associated with. For example, `host`.
TargetName	Name of the target that the alert is associated with. For example, `Database1` or `stadc40.us.oracle.com`.
MetricColumn	Name of the metric that triggered the alert. For example, CPU Utilization(%).

Table 1–3 (Cont.) Enterprise Manager Data Fields

Data Fields	Description
MetricName	Category of the metric. For example, `Load` for the memory utilization alert.
KeyColumn	For metrics that monitor a set of objects, the KeyColumn indicates the type of object monitored. For example, for the Tablespace Space Used (%) metric that monitors tablespace objects, the KeyColumn is 'Tablespace Name'.
KeyValues	Key values associated with a key value base alert. For metrics that monitor a set of objects, KeyValues indicates the specific object that triggered the severity. For example, for the Tablespace Space Used (%) metric that monitors tablespace objects, KeyValues is 'USERS' if the USERS tablespace triggered at warning or critical severity.
Message	Description of the alert. For example, CPU Utilization is 100%, crossed warning (80), or critical (95) threshold.
Severity	Severity of the alert: `critical, warning, clear,` or `down.`
CollectionTime	Timestamp of an alert occurrence.
EventPageURL	Enterprise Manager console URL to the alert details page of the alert.
NotificationRuleName	Name of the notification rule that generated the notification during auto-ticketing.
TargetTimezone	Time zone of the target associated with the alert.
GracePeriodCheckMade	Value `Yes` indicates that the alert is cleared since the last update or creation, but is within the configured grace period.
TargetHost	Name of the server hosting the target that generated the alert.
EventId	Enterprise Manager internal IDs for Target, Metric, and Key.
SeverityCode	Corresponding code for the Severity element. Severity Code has a one-to-one mapping relationship in the order listed. Therefore the first Severity Code entry (ex: <xsd:enumeration value="15"/>) is mapped to the first Severity Element (ex: <xsd:enumeration value="Clear"/>).

Schema

The following schema describes the model that contains the attributes in Table 1–3:

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

  <xsd:element name= "EventModel" type="EMEventModel"/>

  <xsd:complexType name="EMEventModel">
<xsd:sequence>
      <xsd:element name="TicketId" type="xsd:string" minOccurs="0" maxOccurs="1"
/>
      <xsd:element name="ConnectorId" type="xsd:string" minOccurs="1"
maxOccurs="1" />
      <xsd:element name="EventId" type="EventIdType" minOccurs="1" maxOccurs="1"
/>
<xsd:element name="TargetType" type="xsd:string" minOccurs="1" maxOccurs="1" />
```

```
<xsd:element name="TargetName" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="MetricColumn" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="MetricName" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="KeyColumn" type="xsd:string" minOccurs="0" maxOccurs="1" />
<xsd:element name="KeyValues" type="xsd:string" minOccurs="0"
maxOccurs="unbounded" />
<xsd:element name="Message" type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="Severity" type="SeverityType" minOccurs="1" maxOccurs="1" />
<xsd:element name="SeverityCode" type="SeverityCodeType" minOccurs="1"
maxOccurs="1" />
<xsd:element name="CollectionTime" type="xsd:dateTime" minOccurs="1" maxOccurs="1"
/>
<xsd:element name="EventPageURL" type="xsd:string" minOccurs="0" maxOccurs="1" />
      <xsd:element name="EMUser" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="HDUser" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="NotificationRuleName" type="xsd:string" minOccurs="0"
maxOccurs="1" />
      <xsd:element name="TargetHost" type="xsd:string" minOccurs="1" maxOccurs="1"
/>
      <xsd:element name="GracePeriodCheckMade" type="xsd:string" minOccurs="0"
maxOccurs="1" />
      <xsd:element name="TargetTimezone" type="xsd:string" minOccurs="1"
maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

  <xsd:complexType name="EventIdType">
    <xsd:sequence>
      <xsd:element name="TargetId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="MetricId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="KeyId" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

<xsd:simpleType name="SeverityType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="Clear" />
<xsd:enumeration value="Info" />
<xsd:enumeration value="Warning" />
<xsd:enumeration value="Critical" />
<xsd:enumeration value="Agent Unreachable Clear" />
      <xsd:enumeration value="Blackout End" />
      <xsd:enumeration value="Blackout Start" />
      <xsd:enumeration value="Metric Error End" />
      <xsd:enumeration value="Metric Error Start" />
      <xsd:enumeration value="Unknown" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="SeverityCodeType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="15" />
<xsd:enumeration value="18" />
<xsd:enumeration value="20" />
<xsd:enumeration value="25" />
<xsd:enumeration value="115" />
      <xsd:enumeration value="125" />
      <xsd:enumeration value="215" />
      <xsd:enumeration value="225" />
      <xsd:enumeration value="315" />
```

```
      <xsd:enumeration value="325" />
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

Sample Template

Example 1–2 shows a sample ticket template for the Remedy Connector. It creates Remedy tickets with the default categorization `high priority`, and fills the description with alert information.

> **Note:** This sample file is located in the following directory:
>
> `$ORACLE_HOME/sysman/connector/Remedy_Connector`

On updates, the field for work log is updated with a history of the latest changes. The template closes the ticket when the alert clears.

Example 1–2 Sample Ticket Template for Remedy Connector

```
<?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/tt"
                targetNamespace="http://xmlns.oracle.com/sysman/connector/tt"
                elementFormDefault="qualified">
  <!--
  This template creates an incident type ticket with default categorization
  (Category: Default, Type:Default, Item:Default), and high priority. On update,
  the description and message fields are updated, and the ticket is closed if the
  associated alert has cleared.
  -->

  <xsl:template match="ns0:EventModel">
    <xsl:choose>
      <xsl:when test="normalize-space(ns0:TicketId) = ''">
        <urn:Create_Helpdesk_Case xmlns:urn="urn:HelpDesk_Submit_Service">

           <!-- EDIT THE TAG VALUES BELOW TO CHANGE HOW A TICKET IS FILLED
                DURING TICKET CREATION. REFER TO THE REMEDY HELPDESK MANUAL
                FOR DESCRIPTION OF THESE HELPDESK SUPPORT DATAFIELDS-->

        <urn:Case_Type>Incident</urn:Case_Type>
          <urn:Category>Default</urn:Category>
          <urn:Department></urn:Department>
          <urn:Description>
              Ticket created by EM Remedy Connector.
              ------------------------------------
              EM User: <xsl:value-of select="ns0:EMUser"/>

              Event Information:
              Target Type: <xsl:value-of select="ns0:TargetType"/>
              Metric Column: <xsl:value-of select="ns0:MetricColumn"/>
              Metric Name: <xsl:value-of select="ns0:MetricName"/>
              <xsl:choose>
              <xsl:when test="normalize-space(ns0:KeyColumn) != ''">
              Key Column: <xsl:value-of select="ns0:KeyColumn"/>
              Key Values: <xsl:value-of select="ns0:KeyValues"/>
              </xsl:when>
```

```
                </xsl:choose>
                Severity: <xsl:value-of select="ns0:Severity"/>
                Collection Time: <xsl:value-of select="ns0:CollectionTime"/>
                Target Host: <xsl:value-of select="ns0:TargetHost"/>
                <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:EventPageURL"/>
            </urn:Description>
            <urn:Escalated></urn:Escalated>
            <urn:Hotlist></urn:Hotlist>
            <urn:Item>Default</urn:Item>
            <urn:Office></urn:Office>
            <urn:Orig_Submitter>
              <xsl:value-of select="ns0:HDUser"/>
            </urn:Orig_Submitter>
            <urn:Pending></urn:Pending>
            <urn:Phone_Number></urn:Phone_Number>
            <urn:Priority>High</urn:Priority>
            <urn:Region></urn:Region>
            <urn:Request_Urgency>High</urn:Request_Urgency>
            <urn:Requester_Login_Name>
              <xsl:value-of select="ns0:HDUser"/>
            </urn:Requester_Login_Name>
            <urn:Requester_Name>
              <xsl:value-of select="ns0:HDUser"/>
            </urn:Requester_Name>
            <urn:Site></urn:Site>
            <urn:Source>NMP</urn:Source>
            <urn:Status>New</urn:Status>
            <urn:Summary>
              <xsl:value-of select="ns0:Message"/>
            </urn:Summary>
            <urn:Type>Default</urn:Type>
            <urn:WorkLog>
                Severity: <xsl:value-of select="ns0:Severity"/>
                Collection Time: <xsl:value-of select="ns0:CollectionTime"/>
            </urn:WorkLog>
            <urn:Create_Time></urn:Create_Time>
          </urn:Create_Helpdesk_Case>
        </xsl:when>
        <xsl:otherwise>
          <urn:SetBy_Case_ID xmlns:urn="urn:HelpDesk_Modify_Service_w_Wlog">

            <!--  UNCOMMENT THE TAGS YOU WISH TO HAVE MODIFIED WHENEVER
                THE TICKET IS UPDATED, AND GIVE THEM DESIRED VALUES   -->

           <!-- <urn:Accounting_Code></urn:Accounting_Code>           -->
           <!--  <urn:Assignee_Login_Name></urn:Assignee_Login_Name>  -->
           <!--  <urn:Case_Type></urn:Case_Type>                      -->
           <!--  <urn:Category></urn:Category>                        -->
           <!--  <urn:Department></urn:Department>                    -->
                <urn:Description>
                Ticket updated by EM Remedy Connector.
                ---------------------------------
                EM User: <xsl:value-of select="ns0:EMUser"/>
```

```
            Event Information:
            Target Type: <xsl:value-of select="ns0:TargetType"/>
            Metric Column: <xsl:value-of select="ns0:MetricColumn"/>
            Metric Name: <xsl:value-of select="ns0:MetricName"/>
            <xsl:choose>
            <xsl:when test="normalize-space(ns0:KeyColumn) != ''">
            Key Column: <xsl:value-of select="ns0:KeyColumn"/>
            Key Values: <xsl:value-of select="ns0:KeyValues"/>
            </xsl:when>
            </xsl:choose>
            Severity: <xsl:value-of select="ns0:Severity"/>
            Collection Time: <xsl:value-of select="ns0:CollectionTime"/>
            Target Host: <xsl:value-of select="ns0:TargetHost"/>
            <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:EventPageURL"/>
        </urn:Description>
        <!-- <urn:Escalated></urn:Escalated>                      -->
        <!-- <urn:Hotlist></urn:Hotlist>                          -->
        <!-- <urn:Item></urn:Item>                                -->
        <!-- <urn:Office></urn:Office>                            -->
        <!-- <urn:Pending></urn:Pending>                          -->
        <!-- <urn:Phone_Number></urn:Phone_Number>                -->
        <!-- <urn:Priority></urn:Priority>                        -->
        <!-- <urn:Region></urn:Region>                            -->
        <!-- <urn:Request_Urgency></urn:Request_Urgency>          -->
        <!-- <urn:Requester_Login></urn:Requester_Login>          -->
        <!-- <urn:Requester_Name></urn:Requester_Name>            -->
        <!-- <urn:Site></urn:Site>                                -->
        <!-- <urn:Solution_Description></urn:Solution_Description>-->
        <!-- <urn:Solution_Summary></urn:Solution_Summary>        -->
        <!-- <urn:Source></urn:Source>                            -->
        <!-- <urn:Status></urn:Status>                            -->
        <xsl:choose>
         <xsl:when test="ns0:Severity = 'Clear'">
            <urn:Status>Closed</urn:Status>
          </xsl:when>
<xsl:when test="ns0:Severity = 'Agent Unreachable Clear'">
            <urn:Status>Closed</urn:Status>
          </xsl:when>
          <xsl:when test="ns0:GracePeriodCheckMade = 'Yes'">
            <urn:Status>Assigned</urn:Status>
          </xsl:when>
        </xsl:choose>
        <!-- <urn:Submitted_By></urn:Submitted_By>                -->
        <!-- <urn:Summary></urn:Summary>                          -->
        <!-- <urn:Type></urn:Type>                                -->
        <urn:Case_ID>
          <xsl:value-of select="ns0:TicketId"/>
        </urn:Case_ID>
        <urn:WorkLog>
            Severity: <xsl:value-of select="ns0:Severity"/>
            Collection Time: <xsl:value-of select="ns0:CollectionTime"/>
        </urn:WorkLog>
      </urn:SetBy_Case_ID>
    </xsl:otherwise>
</xsl:choose>
```

```
  </xsl:template>
</xsl:transform>
```

Response XSL Files

A response XSL file is needed for the action `Get Ticket`. The file must be named as shown:

```
getTicket_response.xsl
```

> **See Also:** Testing the Help Desk Configuration for details on the Get Ticket operation.

The response XSL file transforms the output from ticketing systems into the format that Enterprise Manager expects, specifically the TicketID.

Example 1–3 shows an example "response transform" for Remedy. It maps the Remedy CaseID ticket attribute to Enterprise Manager's TicketID attribute:

Example 1–3 Example Response Transform for Remedy

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

  <xsl:template match="urn:Create_Helpdesk_CaseResponse">
      <CreateTicketResponse>
        <TicketId><xsl:value-of select="urn:Case_ID"/></TicketId>
      </CreateTicketResponse>
  </xsl:template>
</xsl:transform>
```

Packaging and Deploying the Help Desk Connector

To complete the integration of the connector, package all XML and XSL files (defined in the section Defining XML and XSL Files) as a `.jar` file and then deploy, register, and configure. To do this:

1. Copy the `.jar` file to the host that is running the Oracle Management Service (OMS). For multiple OMSs, you have to copy the `.jar` file for all OMSs.
2. Run the following command on all OMSs:

   emctl extract_jar connector [-jar <jarfile>] [-cname <connectorName>]

   Files are extracted from the `.jar` file to the following directory:

   ```
   $ORACLE_HOME/sysman/connector/connector name
   ```

   > **Note:** *`connector name`* is the name of the connector with spaces (if any) replaced by underscores ("_").

   The full emctl command usage for the extract command is:

   ```
   emctl extract_jar connector [-jar <jarfile>] [-cname <connectorName>]
         -jar          Connector Jar File(full path)
   ```

```
-cname        Connector Name
```

3. Deploy the connector by running the following command:

   emctl register_connector connector [-dd <connectorType.xml>] [-repos_pwd <repos password>]

Example 1–4 Deploying the Help Desk Connector

```
emctl register_connector connector Remedy_Connector.xml  $emHost $emPort $emSID
sysman $sysmanPwd
```

After you deploy a connector, it appears in the Management Connector page (Figure 1–1). After you configure the connector, it is functional.

Note: See also emctl Parameters on page 1-16

The full emctl command usage for the register_connector command is:

```
emctl register_connector connector [-dd <connectorType.xml>] [-repos_pwd <repos
password>]
      -dd           Connector Deployment Descriptor File(full path)
      -repos_pwd    Enterprise Manager Root (SYSMAN) Password
```

4. For every ticket template shipped as part of the connector, run the following command to upload new or modified ticket templates:

   emctl register_template connector [-t <template.xsl>] [-repos_pwd <repos password>] [-ctname <connectorTypeName>] [-cname <connectorName>] [-iname <internalName>] [-tname <templateName>] [-ttype <templateType>] [-d <description>]

Example 1–5 Uploading New or Modified Help Desk Ticket Templates

```
emctl register_ticket_template connector Remedy_DefaultCategory_LowPriority.xsl
$emHost $dbPort $dbSID sysman $sysmanPwd "Remedy Connector" "Remedy Connector"
"Low Priority Template" "This template creates a ticket with low priority and
default categorization"
```

5. Log into Enterprise Manager Console and do the required configurations.

See Also: "Configuring the Help Desk Connector" on page 1-17

The full emctl command usage for the register_template command is

```
emctl register_template connector [-t <template.xsl>] [-repos_pwd <repos
password>] [-ctname <connectorTypeName>] [-cname <connectorName>] [-iname
<internalName>] [-tname <templateName>] [-ttype <templateType>] [-d
<description>]
      -t              Template(full path)
      -repos_pwd      Enterprise Manager Root (SYSMAN) Password
      -ctname         Connector Type Name
      -cname          Connector Name
      -iname          Template Internal Name
      -tname          Template Name Displayed
      -ttype          Template Type
                      <templateType> 1 - inbound transformation
                      <templateType> 2 - outbound transformation
                      <templateType> 3 - XML outbound transformation
```

```
        -d              Description
```

Table 1–4 provides descriptions for the `emctl` parameters.

Table 1–4 emctl Parameters

Parameter	Description
`connectorType.xml`	The connector deployment descriptor.
`ticketTemplate.xsl`	Fully qualified name of the ticket template file. The file resides in the Connector home directory: `$OMS_HOME/sysman/connector/Remedy_Connector` Oracle recommends that you use intuitive names since there might be notification methods created with the same names, and you have to choose one of them when you use the Auto Ticketing feature. Use `xsl` as the file extension since the format is XSLT. For example, `Remedy_DefaultCategory_LowPriority.xsl`. If the file is in a different directory, provide the complete path for the file.
`server`	Host name of the Enterprise Manager repository.
`port`	Listener port of the repository.
`database_sid/ Service Name for RAC DB`	Repository database instance ID or service name if you are using a RAC database as the repository.
`username`	Specify `SYSMAN`.
`password`	Password for `SYSMAN`.
`connectorTypeName`	Connector type name you define in `connectorType.xml`. For example, `"My Ticketing Connector"`. The double quotes ("") are mandatory.
`connectorName`	Connector name. This should be the same as the connector type name. For example, `" My Ticketing Connector"`
`templateName`	An intuitive name for the ticket template that will be displayed in Enterprise Manager.
`description`	A short description for the ticket template. This description is also displayed in Enterprise Manager.
`jar`	Jar file, full jar file path
`cname`	Connector name
`ctname`	Connector type name
`dd`	Connector descriptor file, full path
`repos_pwd`	Enterprise Manager root (SYSMAN) password
`t`	Template file, full path
`iname`	Internal name (For details about this parameter, you can refer to the existing Connector documentation.)
`tname`	Template name
`ttype`	Template type
`d`	Template description

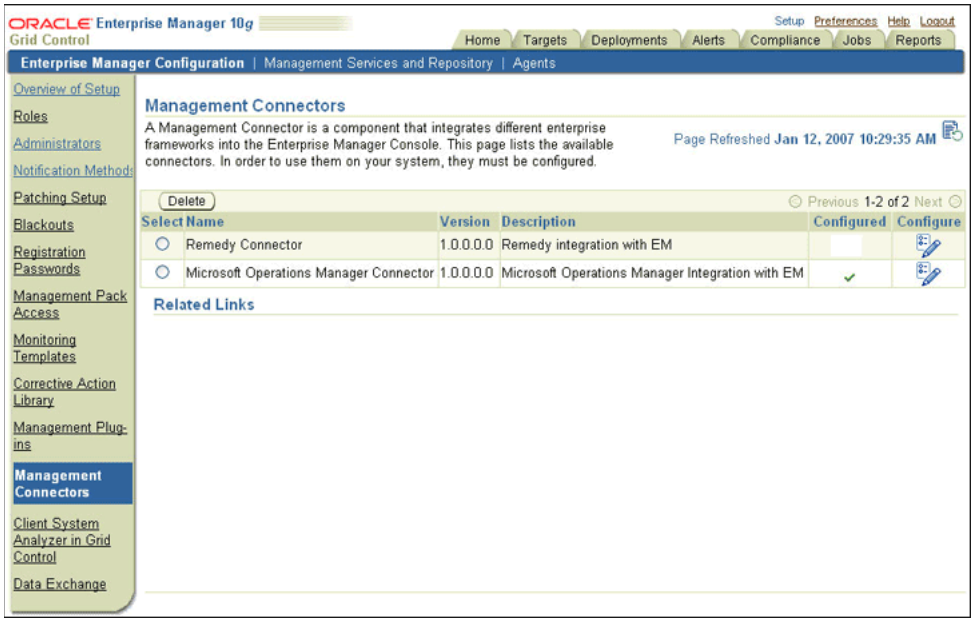
Configuring the Help Desk Connector

To configure the help desk connector:

1. As Super Administrator, log in to the Enterprise Manager console.
2. Click **Setup.**

   The Overview of Setup page appears (Figure 1–1).

Figure 1–1 Management Connectors Page



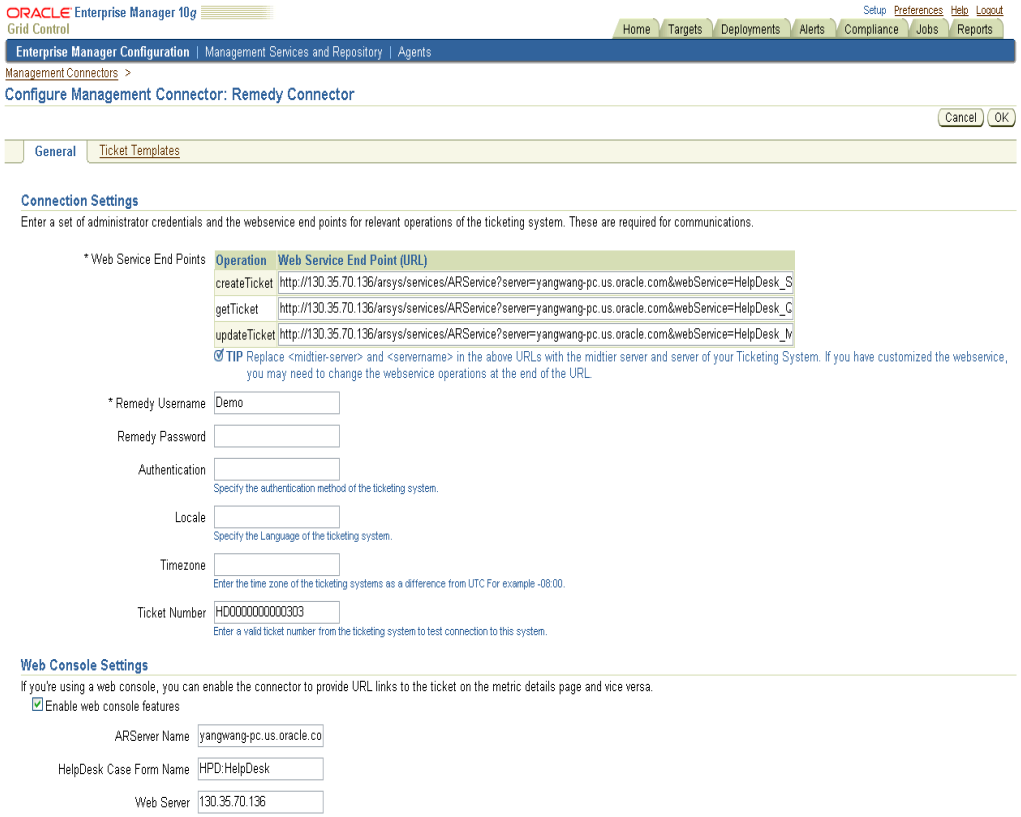
3. Click **Management Connectors** in the left pane.

   The Management Connector Setup page appears. The row for the help desk connector should appear in this page.

4. Click the **Configure** icon for the connector that you registered.

   The General tab of the Configure Management Connector page appears (Figure 1–2).

Figure 1–2 Configure Management Connector Page for Remedy Connector



5. Configure the following:
   - Connection settings
     - WS End Points
     - Related Fields
   - Web Console settings
   - Grace Period
6. Click **OK.** The Management Connector Setup page reappears. The help desk connector row should have a check mark in the Configured column.
7. In the Configure Management Connector page, go to the Ticket Templates tab and ensure that ticket templates are successfully loaded.

For more detailed configuration information, see the associated chapter for your particular help desk connector, such as Remedy or Siebel, in the *Oracle Enterprise Manager Connector Installation and Configuration Guide*.

Testing the Help Desk Configuration

Rather than saving your configuration without knowing whether it is correct as intended, you can use the Ticket Number field in Table 1–2 to test the connection to this system. To customize the test to suit your needs, Oracle has provided the required

default Remedy files that you can modify for other help desk ticketing systems, such as Siebel. The basic process to use these files is as follows:

1. Modify the transformation file (XSLT) to transform the generic `getTicket` request into connector-compliant XML data.
2. Modify the transformation file (XSLT) to transform the connector-compliant `getTicket` response back to the generic XML data.

The following sections explain the files and code used for this process.

Transforming the Request into Connector-compliant XML Data

The generic `getTicket` request is defined in the `getTicket_request.xsd` XML schema, which is located here:

`$ORACLE_HOME/sysman/connector/common/schema`

The content of this file is shown below:

```
<?xml version="1.0" encoding="US-ASCII" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="getTicketRequest">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="ticketID" type="xs:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

According to this schema, a sample generic `getTicket` request for any help desk connector is as follows:

```
<getTicketRequest>
     <ticketID> … </ticketID>
</getTicketRequest>
```

This is generated by the Management Connector Framework, which an XSLT transformation file, `getTicket_request.xsl`, needs to transform into connector-compliant XML data. The transformation file for Remedy Connector is located here:

`$ORACLE_HOME/sysman/connector/<Remedy_Connector>/getTicket_request.xsl`

The content of this XSLT file is shown below:

```
<?xml version='1.0' encoding='UTF-8'?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="getTicketRequest">
        <urn:GetListBy_Case_ID xmlns:urn="urn:HelpDesk_Query_Service">
                <urn:Case_ID>
                        <xsl:value-of select="ticketID/text()"/>
                </urn:Case_ID>
        </urn:GetListBy_Case_ID>
    </xsl:template>
</xsl:transform>
```

For help desk connectors other than Remedy, you need to modify this XSLT file and place it in a similar location as shown for the Remedy Connector above. The file

should transform the generic `getTicket` request to data acceptable to the ticketing system, as shown below for a Remedy system:

```
<urn:GetListBy_Case_ID xmlns:urn="urn:HelpDesk_Query_Service" >
        <urn:Case_ID>HD0000000016686</urn:Case_ID>
</urn:GetListBy_Case_ID>
```

Transforming the Response Back to the Generic XML Data

The generic `getTicket` response that the management connector framework can accept is defined by the `getTicket_response.xsd` XML schema, which is located here:

`$ORACLE_HOME/sysman/connector/common/schema`

The content of this file is shown below:

```
<?xml version="1.0" encoding="US-ASCII" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="getTicketResponse">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="ticketID" type="xs:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

According to this schema, a sample generic `getTicket` response for any help desk connector is as follows:

```
<getTicketResponse>
          <ticketID> … </ticketID>
</getTicketResponse>
```

To transform the connector-compliant `getTicket` response back to the generic XML data, you need to use a the `getTicket_response.xsl` transformation file. For the Remedy connector, the file is located here:

`$ORACLE_HOME/sysman/connector/<Remedy_Connector>/getTicket_response.xsl`

The content of this XSLT file for the Remedy Connector is shown below:

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

You need to modify this `getTicket_response.xsl` file for help desk connectors other than Remedy and place it in a similar location as shown for the Remedy Connector above.

Enabling an SSL Connection for HTTPS

Follow the steps provided in this section if you choose HTTPS as the protocol to establish a connection between the external ticketing system and Enterprise Manager.

Generating Certificate Request File

Generate a certificate request file for the external ticketing system and send it to the Certificate authority, such as VeriSign.

> **Note:** The certificate request file is dependent on the Web server used by the external ticketing system.

Importing the Certificate from the Certificate Authority

After you get the certificate, import it to the Web server the external ticketing system uses. The import mechanism varies depending on the Web server the external ticketing system uses.

Adding Signed Certificates to Wallet Manager

> **Note:** Oracle Wallet Manager is available at `$ORACLE_HOME/bin` on OMS. See the *Oracle Application Server Administrator's Guide* for details.

1. Get the port number from *user_projects/domains/EMGC_DOMAIN/config/config.xml*:

```
<server>
<name>EMGC_ADMINSERVER</name>
<ssl>
<name>EMGC_ADMINSERVER</name>
<enabled>true</enabled>
<hostname-verification-ignored>true</hostname-verification-ignored>
<listen-port>7022</listen-port>
</ssl>
<listen-port-enabled>false</listen-port-enabled>
<listen-address>server_name</listen-address>
</server>
```

2. Connect to the WebLogic Admin Console:
   a. Connect to *https://server_name:7022/console/*, login: [username]/[password]
   b. Navigate to *Environment->Servers->EMGC_DOMAIN->Keystore tab*
   c. If two-way SSL is required for external web service, go to *SSL->Advanced->Use Server Certs* and be sure it is checked
   d. Make sure that *CA signed the external system server's certificate* is added to trustStore specified in the Keystore tab, for example, *WebLogic's DemoTrust*, or *java standard cacerts*

> **See Also:** For information on creating an Oracle Wallet, see "Creating and Viewing Oracle Wallets with orapki" in the *Oracle Database Advanced Security Administrator's Guide, 10g Release 2 (10.2).*

2

Building an Event Connector

This chapter provides information that you need to build an event connector and integrate it with Enterprise Manager.

This chapter has the following sections:

Introduction

Enterprise Manager 11*g* provides a Management Connector Framework (referred to as Connector Framework) that enables developers to build event connectors based on metadata (XMLs and XSLs). The event connector allows Enterprise Manager to retrieve external events using a poll-based approach. You can also send events to external systems.

The following categories of metadata files are required to build an event connector:

- Connector deployment descriptor

  If `IsNewTargetType` is set to `true`, target type (`targetType.xml`) and default target (`defaultTargetInstance.xml`) definition files are also required.

  > **Note:** For Enterprise Manager 10g Release 4, you must set `IsNewTargetType` to `true`.

- Request transformation XSLs
- Response transformation XSLs
- Request XMLs

Prerequisites

Before building the connector framework, ensure that you meet the following event system prerequisites:

- The external event system exposes the following Web services:
  - getNewAlerts
  - getUpdatedAlerts
  - acknowledgAlerts (optional)
  - getResponse (optional)
  - updateAlert (optional)
  - uninitialize (optional)
  - initialize (optional)
  - setup (optional)
  - createEvent (optional)
  - updateEvent (optional)
  - cleanup (optional)

  Note: The custom Web service endpoint names can differ from those listed, and they are mapped by the connector framework.

- The Web service is of Document Literal style
- Web services support SOAP header-based authentication.

Defining XML and XSL Files

The following sections provide details of various XML and XSL files that you must define for building an event connector.

Oracle recommends that you refer to the metadata files at the following location:

$OMS_HOME/sysman/connector/Microsoft_Operations_Manager_Connector

Connector Descriptor XML File

Define a connector descriptor XML file to describe the connector metadata and the configuration properties of the connector, such as Web service end points and authentication schema. This file should adhere to the schema `connectorType.xsd`.

A sample connector descriptor XML file is provided below. You need to define various elements. To do this, replace the highlighted text with the corresponding elements.

Example 2–1 Sample Connector Descriptor XML File

```
<?xml version='1.0' encoding='UTF-8'?>
<ManagementConnector xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     xmlns="http://xmlns.oracle.com/sysman/connector">
  <Name>specify the connector name</Name>
  <Version>specify the 5-digit version, e.g., 1.0.0.0.0</Version>
  <Description>specify a short description</Description>
  <Category>EventConnector</Category>
```

```
<!-- Category is fixed to EventConnector. -->
  <EventConnector>
    <!-- If IsNewTargetType is set to true, it means the developers want to define
a new target type to hold external alerts. In that case, targetType.xml and
defaultTargetType.xml need to be shipped in the connector jar. -->
    <IsNewTargetType>true</IsNewTargetType>
    <EventService>
      <Method>getNewAlerts</Method>
      <WebServiceEndpoint>
        <![CDATA[specify URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>getUpdatedAlerts</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
<Method>acknowledgeAlerts</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>setup</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>destroy</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
   <!-- BaseURL element is for the default UI for event connectors, the following
value should not be changed. -->
    <BaseURL>/em/console/connector/event/configuration</BaseURL>
  </EventConnector>
</ManagementConnector>
```

connectorType.xsd File

```
<?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="ManagementConnector">
    <xsd:annotation>
      <xsd:documentation>Deployment Descriptor for Management
Connectors</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="Version" type="xsd:string"/>
        <xsd:element name="Description" type="xsd:string"/>
        <xsd:element name="Category">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="EventConnector"/>
              <xsd:enumeration value="TicketingConnector"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:choice>
          <xsd:element name="EventConnector" type="EventConnector"/>
          <xsd:element name="TicketingConnector" type="TicketingConnector"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="EventConnector">
    <xsd:sequence>
      <xsd:element name="IsNewTargetType" type="xsd:boolean"/>
      <xsd:element name="Authentication" type="Authentication" minOccurs="0"/>
      <xsd:element name="HTTPAuthentication" type="Authentication" minOccurs="0"/>
      <xsd:element name="ConfigData" type="ConfigData" minOccurs="0"/>
      <xsd:element name="EventService" type="EventService" minOccurs="1"
maxOccurs="unbounded"/>
      <xsd:element name="TargetResponse" type="Response" minOccurs="0"/>
      <xsd:element name="BaseURL" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TicketingConnector">
    <xsd:sequence>
      <xsd:element name="Authentication" type="Authentication" minOccurs="0"/>
      <xsd:element name="HTTPAuthentication" type="Authentication" minOccurs="0"/>
      <xsd:element name="TicketingService" type="TicketingService" minOccurs="1"
maxOccurs="unbounded"/>
      <xsd:element name="BaseURL" type="xsd:string"/>
      <xsd:element name="ExternalURL" type="ExternalURL"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="EventService">
    <xsd:sequence>
      <xsd:element name="Method" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="getNewAlerts"/>
            <xsd:enumeration value="getUpdatedAlerts"/>
            <xsd:enumeration value="acknowledgeAlerts"/>
            <xsd:enumeration value="setup"/>
```

```
            <xsd:enumeration value="initialize"/>
            <xsd:enumeration value="uninitialize"/>
            <xsd:enumeration value="cleanup"/>
            <xsd:enumeration value="updateAlerts"/>
            <xsd:enumeration value="createEvent"/>
            <xsd:enumeration value="updateEvent"/>
            <xsd:enumeration value="getResponse"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="WebServiceEndpoint" type="xsd:string"/>
      <xsd:element name="SOAPAction" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TicketingService">
    <xsd:sequence>
      <xsd:element name="Method" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="createTicket"/>
            <xsd:enumeration value="updateTicket"/>
            <xsd:enumeration value="getTicket"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="WebServiceEndpoint" type="xsd:string"/>
      <xsd:element name="SOAPAction" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Authentication">
    <xsd:sequence>
      <xsd:element name="Username" type="Username"/>
      <xsd:element name="Password" type="Password"/>
      <xsd:element name="Attribute" type="Attribute" maxOccurs="unbounded"
minOccurs="0"/>
      <xsd:element name="Data" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ConfigData">
    <xsd:sequence>
      <xsd:element name="ConfigProperty" type="ConfigProperty"
maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Username">
    <xsd:sequence>
      <xsd:element name="AttributeName" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Password">
    <xsd:sequence>
      <xsd:element name="AttributeName" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Attribute">
    <xsd:sequence>
```

```
      <xsd:element name="AttributeName" type="xsd:string"/>
      <xsd:element name="DisplayName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>


  <xsd:complexType name="ConfigProperty">
    <xsd:sequence>
      <xsd:element name="PropertyName" type="xsd:string"/>
      <xsd:element name="DisplayValue" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Response">
    <xsd:sequence>
      <xsd:element name="ScriptName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ExternalURL">
    <xsd:sequence>
      <xsd:element name="Pattern" type="xsd:string"/>
      <xsd:element name="UserVariable" type="xsd:string" maxOccurs="unbounded"
minOccurs="0"/>
      <xsd:element name="SubstitutionVariable" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Target Type Definition XML File

The target type definition XML file defines the target type where the retrieved external events are stored; for example, `mom_managed_host` for alerts from MOM. The file must be named `targetType.xml`.

targetType.xml File Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<TargetMetadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               TYPE="hp_managed_host" TYPEDISPLAYNAME="OVO Managed Host" >
 <Metrics>
  <Metric>
    <MetricName>
        Response
    </MetricName>
    <Display>Response</Display>
    <Category>Status</Category>
    <ProducerId>xyz</ProducerId>
    <MetricColumns>
     <Column>Status</Column>
    </MetricColumns>
    <!-- No other Metric Columns should be defined for response Metric -->
  </Metric>
         <Metric>
   <MetricName>Generic_Alert</MetricName>
   <Display>Generic_Alert</Display>
   <Category>Generic_Alert</Category>
   <ProducerId>xyz</ProducerId>
   <MetricColumns>
```

```
    <Column1>AlertId</Column1>
    <Column2>ComputerName</Column2>
    <Column3>DomainName</Column3>
    <Column4>Source</Column4>
    <Column5>Rule</Column5>
   </MetricColumns>
  </Metric>
 </Metrics>
  <InstanceProperties>
    <InstanceProperty NAME="HOSTNAME" CREDENTIAL="FALSE" OPTIONAL="FALSE">
      <Display>
        <Label>HOSTNAME_DISPLAY</Label>
        <ShortName>HOSTNAME</ShortName>
      </Display>
    </InstanceProperty>
  </InstanceProperties>
</TargetMetadata>
```

The target type definition is loaded during the deployment of the connector.

Adding More Properties

The `targetType.xml` file defines instance properties, and you can add multiple instance properties as in the following example:

```
<InstanceProperty NAME="HOSTNAME" CREDENTIAL="FALSE" OPTIONAL="FALSE">
   <Display>
    <Label>HOSTNAME</Label>
    <ShortName>HOSTNAME</ShortName>
   </Display>
  </InstanceProperty>
```

The instance properties should be the identifying attributes for the targets. For example, if hostname is sufficient to uniquely identify the target, there should be only one instance property, `hostname`.

Defining Metrics in Enterprise Manager

This section explains how you define and manage metrics in Enterprise Manager.

In the `targetType.xml` file, define at least two metrics for a new target type. The two mandatory metrics are:

- `Response`
- `Generic_Alert`

For example:

```
<Metrics>
  <Metric>
    <MetricName>Response</MetricName>
    <Display>Response</Display>
    <Category>Status</Category>
    <ProducerId>MySystem</ProducerId>
    <MetricColumns>
     <Column>Status</Column>
    </MetricColumns>
    <!-- No other Metric Columns should be defined for response Metric -->
  </Metric>
  <Metric>
   <MetricName>Generic_Alert</MetricName>
```

```
   <Display>Generic_Alert</Display>
   <Category>Generic_Alert</Category>
   <ProducerId>MySystem</ProducerId>
   <MetricColumns>
    <Column1>AlertId</Column1>
   </MetricColumns>
  </Metric>
</Metrics>
```

When external alerts are retrieved, they are mapped to a metric (`<MetricName/>` element in the response XSLT). If the metric is already defined, the alerts are associated with them. If not, a new metric is instantly created using `Generic_Alert` as a template, which is structurally the same as `Generic_Alert` but with a different name.

A maximum of 100 metrics are instantly created. Beyond that, alerts (that cannot be associated with existing metrics) are associated with `Generic_Alert`.

You can have up to five key fields as described in the sample below. One key field (equivalent of Alert ID) is mandatory. Column 1 identifies the uniqueness of the alert. Other columns are optional.

`<Column1>AlertId</Column1>`

`<Column2>sth</Column2>`

`<Column3>customfield1</Column3>`

`<Column4>ruleid</Column4>`

`<Column5>description</Column5>`

Ensure that the key fields you define in `getNewAlerts_response.xsl` and `getUpdatedAlerts_response.xsl` are populated with appropriate values.

For example:

```
<key1>
   <xsl:value-of select="a:AlertId"/>
  </key1>
<key2>
   <xsl:value-of select="a:sth"/>
  </key1>
<key3>
   <xsl:value-of select="a:customfield1"/>
  </key3>
<key4>
   <xsl:value-of select="a:ruleId"/>
  </key4>
<key5>
   <xsl:value-of select="a:description"/>
  </key5>
```

Define only the key fields you defined for the `Generic_Alert` metric. For example, if you define only one key field for `Generic_Alert`, you need only transform the `<key1/>` element. If any extra values are added in the XSLT, they are removed during the alert processing.

If the key/field is declared and no value is assigned, a space (" ") is automatically added to the key/field so that the alert can be inserted into Enterprise Manager.

Generic Target Instance Definition XML File

Provide a default target instance definition based on the target type you define. Name the file as `defaultTargetInstance.xml`. The file is read at the time you deploy the connector and a default target instance is created.

Name the default target instance as follows:

`generic_<target_type_name>`

For example, the target type defined for MOM Connector is `mom_managed_host` and the default target instance is `generic_mom_managed_host`.

By default, the default target instance holds all incoming alerts if no other specific target instance is created.

If the target type already exists in Enterprise Manager and you try to recreate it, the connector is not re-registered. Therefore, the configuration changes that you expect will not occur.

Default Request XML Files

The Request XML and XSL files are required for a successful Web service operation. For example, the Microsoft Operations Manager (MOM) Connector needs the setup name and resolution state to be part of the setup Web service call to MOM. The request file, `setup_request.xml` is used for this.

Table 2–1 lists the Request XML files that you need to create. Ensure that the file names you provide exactly match those provided in the table.

Table 2–1 Default Request XML Files

Template	Description
getNewAlerts_request.xml (optional *)	Generates the initial request XML file for new alerts.
updateAlerts_request.xml	Generates the initial request XML file for update alerts.
setup_request.xml	Registers the connector with the external eventing system.
initialize_request.xml	Initializes the connector with the external eventing system after setup.
uninitialize_request.xml	Un-initializes the connector with the external eventing system.
cleanup_request.xml	De-registers the connector with the external eventing system.
acknowledgeAlerts_request.xml	Defines the acknowledgement request and sends it to the external eventing system.

*Either *getNewAlerts_request.xml* or *getNewAlerts_request.xsl* should be registered for *getNewAlerts*. Either *getUpdatedAlerts_request.xml* or *getUpdatedAlerts_request.xsl* should be registered for *getUpdatedAlerts*.

** *CreateEvent* and *updateEvent* are optional. But they are required if user wants to forward alert from EM to external system.

The following details apply to all methods in Table 2–2:

- Setup and initialize are called in the order when the connector is configured if they're defined.

- Uninitialize and cleanup are opposite actions to initialize and setup. uninitialize can be defined if initialize is defined. cleanup can be defined if setup is defined. They're called in the order of uninitialize and cleanup when the connector is deleted if they're defined.
- GetNewAlerts and getUpdatedAlerts are called in the order when the EventCollection job is run every polling interval. For each new/updated alert, after it's been inserted/updated in Enterprise Manager, acknowledgeAlert and updateAlert will be called if they are defined.
- CreateEvent and updateEvent are called to forward alert from EM to external system through notification. They're optional but need to be defined if you want this functionality.
- GetResponse is called after getNewAlerts and getUpdatedAlerts when the EventCollection job is run every polling interval. It's used to update the status of proxy target added for the connector. This method is optional.

The following details apply to all templates in Table 2–2:

- From version 10.2.0.5, the templates will be registered and saved in the repository. When registering the template, the internal name (-iname) should be the same as the method name (case sensitive), the template type (-ttype) shoud be: 1 for response xsl, 2 for request xsl, and 3 for request xml. For example, getNewAlerts_response.xsl should be registered with the internal name getNewAlerts and template type 1.
- The template filename is no longer important, but we recommend using <methodName>_request.xml, <methodName>_request.xsl, and <methodName>_response.xsl.

The following examples show the code for these files.

generic_request_newalerts.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<getAlerts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.oracle.com/sysman/connector
GenericGetAlertRequest.xsd"
        xmlns="http://xmlns.oracle.com/sysman/connector">
   <registrationId>$REGISTRATION_ID$</registrationId>
   <maxCount>100</maxCount>
   <typeOfAlerts> NewAlerts</typeOfAlerts>
   <getAlertsData>
      <ConfigParam>
        <Name></Name>
        <Value></Value>
    </ConfigParam>
   </getAlertsData>
</getAlerts
```

generic_request_updatedalerts.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<getAlerts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.oracle.com/sysman/connector
GenericGetUpdateRequest.xsd"
        xmlns="http://xmlns.oracle.com/sysman/connector">
   <registrationId>$REGISTRATION_ID$</registrationId>
   <maxCount>100</maxCount>
   <typeOfAlerts> UpdatedAlerts</typeOfAlerts>
   <getAlertsData>
```

```
      <ConfigParam>
        <Name></Name>
        <Value></Value>
    </ConfigParam>
   </getAlertsData>
</getAlerts
```

setup_request.xml (optional)

```
<?xml version="1.0" encoding="UTF-8" ?>
<setup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.oracle.com/sysman/connector
GenericSetupRequest.xsd"
       xmlns="http://xmlns.oracle.com/sysman/connector">
  <setupname>$SETUP_NAME$</setupname>
  <setupData>
    <ConfigParam>
      <Name></Name>
      <Value></Value>
    </ConfigParam>
  </setupData>
</setup
```

initialize_request.xml (optional)

```
<?xml version="1.0" encoding="UTF-8" ?>
<initialize xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://xmlns.oracle.com/sysman/connector">
  <registrationId>$REGISTRATION_ID$</registrationId>
  <typeofAlerts>NewAlerts UpdatedAlerts</typeofAlerts>
</initialize>
```

uninitialize_request.xml (optional)

```
<?xml version="1.0" encoding="UTF-8" ?>
<uninitialize xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://xmlns.oracle.com/sysman/connector">
  <registrationId>$REGISTRATION_ID$</registrationId>
  <typeofAlerts>NewAlerts UpdatedAlerts</typeofAlerts>
</uninitialize>
```

cleanup_request.xml (optional)

```
<?xml version="1.0" encoding="UTF-8" ?>
<cleanup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://xmlns.oracle.com/sysman/connector">
  <registrationId>$REGISTRATION_ID$</registrationId>
  <cleanupData>
    <ConfigParam>
      <Name></Name>
      <Value></Value>
    </ConfigParam>
  </cleanupData>
</cleanup>
```

acknowledge_request.xml (optional)

```
<?xml version="1.0" encoding="utf-8" ?>
<getAlerts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.oracle.com/sysman/connector
GenericGetAlertRequest.xsd"
       xmlns="http://xmlns.oracle.com/sysman/connector">
  <registrationId>$REGISTRATION_ID$</registrationId>
```

```
  <maxCount>100</maxCount>
  <typeOfAlerts> NewAlerts</typeOfAlerts>
  <getAlertsData>
     <ConfigParam>
       <Name></Name>
       <Value></Value>
   </ConfigParam>
  </getAlertsData>
</getAlerts>
```

Default Request XSL Files

Table 2–2 provides the list of request XSL files that you must define for building an event connector. If no transformation is required, you can define the following XSL files to allow the contents to pass through without any modification.

Table 2–2 Default Request XSL Files

File Name	Description
setup_request.xsl	Transforms `setup_request.xml`.
initialize_request.xsl	Transforms `initialize_request.xml`.
getNewAlert_request.xsl	Transforms `generic_request_newalerts.xml`.
getUpdatedAlert_request.xsl	Transforms `generic_request_updatedalerts.xml`.
acknowledge_request.xsl	Transforms `generic_request_acknowledgealerts.xml`.
uninitialize_request.xsl	Transforms `uninitialize_request.xml`.
cleanup_request.xsl	Transforms `cleanup_request.xml`.

Response XSL Files

Response XSL files transform response received from the external event system. The response can be setup response, initialize response, and so on. They may not be related to alerts from external systems.

Table 2–3 lists the Response XSL files you need to create. Ensure that the file names you provide exactly match those provided in the table.

Table 2–3 Response XSL Files

File Name	Description

setup_response.xsl (optional)	Converts the incoming response from the external eventing system for Enterprise Manager to understand the registration ID.
	Subsequently, the external system uses this ID to identify the event connector.
	The response from the external system should be converted to the following format so that Enterprise Manager can understand and add the response to its repository:
	`<registrationId>`
	`122222`
	`</registrationId>`

getNewAlerts_response.xsl	Transforms the new incoming alerts to the base Enterprise Manager alert schema.

Table 2–3 (Cont.) Response XSL Files

File Name	Description
getUpdatedAlerts_response.xsl	Transforms the incoming updated alerts to the base Enterprise Manager alert schema.

The following examples show the code for these files.

setup_response.xsl (optional)

```
<?xml version='1.0' ?>
<xsl:stylesheet version="1.0"
    xmlns:ns0="http://www.microsoft.com/EnterpriseManagement/Mom/Connector/V2"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:a="http://xmlns.oracle.com/sysman/connector">
        <xsl:template match="/">
          <ns0:registrationId>
                <xsl:value-of
select="ns0:SetupWithResolutionStateResponse/ns0:SetupWithResolutionStateResult"/>
           </ns0:registrationId>
        </xsl:template>
</xsl:stylesheet>
```

getNewAlerts_response.xsl

```
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<EMAlerts>
        <xsl:for-each select="getNewAlertsReturn/item">
        <Alert>
                <key1>
                        <xsl:value-of select="id"/>
                </key1>
                <key2>
                        <xsl:value-of select="application"/>
                </key2>
                <key3>
                        <xsl:value-of select="messageGroup"/>
                </key3>
                <key4>
                        <xsl:value-of select="node"/>
                </key4>
                <key5>
                     <xsl:value-of select="serviceName"/>
                </key5>
                <message>
                        <xsl:value-of select="messageText"/>
                </message>
                <producerID>xyz</producerID>
                <targetName>
                        <xsl:value-of select="node"/>
                </targetName>
                <targetType>hp_managed_host</targetType>
                <timeStamp>2006-10-09T09:23:32.6430000-07:00</timeStamp>
                <metricName>
                        <xsl:value-of select="source"/>
                </metricName>
                <category>
```

```
                        <xsl:value-of select="messageGroup"/>
                </category>
                <severity>
                        <xsl:variable name="ovoseverity"><xsl:value-of
select="severity"/></xsl:variable>
                        <xsl:variable name="critical">32</xsl:variable>
                        <xsl:variable name="major">128</xsl:variable>
                        <xsl:variable name="warning">16</xsl:variable>
                        <xsl:variable name="minor">64</xsl:variable>
                        <xsl:variable name="normal">8</xsl:variable>
                        <xsl:variable name="unknown">4</xsl:variable>
                        <xsl:variable name="none">0</xsl:variable>
                        <xsl:variable name="emclear">Clear</xsl:variable>
                        <xsl:variable name="emcritical">Critical</xsl:variable>
                        <xsl:variable name="emwarning">Warning</xsl:variable>
                        <xsl:variable
name="eminformational">Informational</xsl:variable>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $critical">
                        <xsl:value-of select="$emcritical"/>
                        </xsl:when>
                        </xsl:choose>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $unknown">
                        <xsl:value-of select="$eminformational"/>
                        </xsl:when>
                        </xsl:choose>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $none">
                       <xsl:value-of select="$eminformational"/>
                        </xsl:when>
                        </xsl:choose>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $warning">
                        <xsl:value-of select="$emwarning"/>
                        </xsl:when>
                        </xsl:choose>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $major">
                        <xsl:value-of select="$emcritical"/>
                        </xsl:when>
                        </xsl:choose>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $minor">
                        <xsl:value-of select="$eminformational"/>
                        </xsl:when>
                        </xsl:choose>
                        <xsl:choose>
                        <xsl:when test="$ovoseverity = $normal">
                        <xsl:value-of select="$emclear"/>
                        </xsl:when>
                        </xsl:choose>
                </severity>
                </Alert>
                </xsl:for-each>
        </EMAlerts>
        </xsl:template>
</xsl:stylesheet>
```

getUpdatedAlerts_response.xsl

```
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns:a="http://www.microsoft.com/EnterpriseManagement/Mom/Connector/V2">
 <xsl:template match="/">
<EMAlerts>
  <xsl:for-each
select="a:GetDataResponse/a:GetDataResult/a:UpdatedAlerts/a:Alert">
   <Alert>
    <key1>
     <xsl:value-of select="a:AlertId"/>
    </key1>
     <key2>
       <xsl:value-of select="a:ComputerName"/>
     </key2>
     <key3>
       <xsl:value-of select="a:ComputerDomain"/>
     </key3>
     <key4>
       <xsl:value-of select="a:Name"/>
     </key4>
     <key5>
       <xsl:value-of select="a:RuleId"/>
     </key5>
   <message>
       <xsl:value-of select="a:Description"/>
   </message>
   <producerID>xyz</producerID>
   <targetName>
    <xsl:value-of select="a:ComputerName"/>
   </targetName>
   <targetType>hp_managed_host</targetType>
   <timeStamp>
    <xsl:value-of select="a:TimeLastModified"/>
   </timeStamp>
   <username></username>
   <password></password>
   <metricName>
        <xsl:value-of select="a:Source"/>
   </metricName>
   <category>
           <xsl:value-of select="a:Source"/>
   </category>
    <value>
          <xsl:value-of select="a:sth"/>
    </value>
 <severity>
     <xsl:variable name="momseverity" select="a:Severity"/>
     <xsl:variable name="criticalerror">CriticalError</xsl:variable>
     <xsl:variable name="error">Error</xsl:variable>
     <xsl:variable name="warning">Warning</xsl:variable>
     <xsl:variable name="information">Information</xsl:variable>
     <xsl:variable name="securityissue">SecurityIssue</xsl:variable>
     <xsl:variable name="unknown">Unknown</xsl:variable>
     <xsl:variable name="serviceunavailable">ServiceUnavailable</xsl:variable>
     <xsl:variable name="success">Success</xsl:variable>
     <xsl:variable name="emclear">Clear</xsl:variable>
     <xsl:variable name="emcritical">Critical</xsl:variable>
```

```
    <xsl:variable name="emwarning">Warning</xsl:variable>
    <xsl:variable name="eminformational">Informational</xsl:variable>
    <xsl:choose>
            <xsl:when test="$momseverity = $criticalerror">
             <xsl:value-of select="$emcritical"/>
            </xsl:when>
    </xsl:choose>
    <xsl:choose>
            <xsl:when test="$momseverity = $unknown">
             <xsl:value-of select="$eminformational"/>
            </xsl:when>
    </xsl:choose>
     <xsl:choose>
            <xsl:when test="$momseverity = $error">
             <xsl:value-of select="$emcritical"/>
            </xsl:when>
    </xsl:choose>
     <xsl:choose>
            <xsl:when test="$momseverity = $securityissue">
             <xsl:value-of select="$emcritical"/>
            </xsl:when>
    </xsl:choose>
    <xsl:choose>
            <xsl:when test="$momseverity = $warning">
             <xsl:value-of select="$emwarning"/>
            </xsl:when>
    </xsl:choose>
    <xsl:choose>
            <xsl:when test="$momseverity = $serviceunavailable">
             <xsl:value-of select="$eminformational"/>
            </xsl:when>
    </xsl:choose>
    <xsl:choose>
            <xsl:when test="$momseverity = $information">
            <xsl:value-of select="$eminformational"/>
            </xsl:when>
     </xsl:choose>
    <xsl:choose>
            <xsl:when test="$momseverity = $success">
             <xsl:value-of select="$emclear"/>
            </xsl:when>
     </xsl:choose>
   </severity>
   </Alert>
  </xsl:for-each>
  </EMAlerts>
 </xsl:template>
</xsl:stylesheet>
```

Response Metric Collection Script

The `getResponse` operation is mapped to a Web service. The connector framework uses it to determine the response of the existing target in Enterprise Manager for the defined target type.

Table 2–4 provides the metadata details. The required meta files are similar to any other methods described in "Default Request XML Files" on page 2-9.

Note: The Response Metric Collection script does not apply for building the MOM Connector.

Table 2–4 Metadata for getResponse Operation

Value	Description
getResponse_request.xml	Template to generate the `getResponse` request.
getResponse_request.xsl	Transforms `getResponse_request.xml`.

The status returned should be transformed into the following status codes:

0 — Target down
1 — Target up

If neither value applies, the target status displays as `Pending`.

To determine the response status, Enterprise Manager connector sends `getResponse_request.xml` in the following format:

```
<?xml version="1.0" encoding="utf-8" ?>
<GetResponseRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Target>
             <Name>SMP-MPI2</Name>
             <InstanceProperty>
                      <Name>host name</Name>
                      <Value>smp-mpi2.us.oracle.com</Value>
             </InstanceProperty>
              <!-- InstanceProperty
                          can repeat multiple times. InstanceProperty are the
                          properties that can identify the managed entity in the
external system. -->
      </Target>
     <!-- Target element can repeat. -->
     <Attribute>
            <Name>RegistrationId</Name>
            <Value>$REGISTRATION_ID$</Value>
               <!-- one example is RESITRATION_ID,
                       <Name>registrationId</Name><Value>$REGISTRATION_ID$</Value>
               -->
     </Attribute>
</GetResponseRequest>
```

In this case, you have to define an XSL file named `getResponse_request.xsl` to compute and return a response to Enterprise Manager in the following XML format that Enterprise Manager can understand:

```
<?xml version="1.0" encoding="utf-8" ?>
<GetResponseReponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Target>
             <Name>SMP-MPI2</Name>
             <Type>hp_managed_node</Type>
             <Status>0<Status>
      </Target>
</GetResponseResponse>
```

Based on the response above, the status of the target is updated in Enterprise Manager.

Packaging and Deploying the Connector

To complete the integration of the connector, package all XML and XSL files (defined in the section "Defining XML and XSL Files") as a `.jar` file and then deploy, register, and configure the connector. To do this:

1. Copy the `.jar` file to the Oracle Management Service (OMS). For multiple OMSs, copy the `.jar` file for all OMSs.
2. Run the following command on all OMS's:

   ```
   emctl extract_jar connector [-jar <jarfile>] [-cname
   <connectorName>]
   ```

   Where `-jar` is the Connector Jar File(full path) and `-cname` is the Connector Name

   Files are extracted from the `.jar` file to the following directory:

   `$ORACLE_HOME/sysman/connector/`*`connector name_wo_space`*

   The command replaces the spaces in the `connector_name` with an underscore ("_") in `connector_name_wo_space`.
3. Deploy the connector by running the following command:

   ```
   emctl register_connector connector [-dd <connectorType.xml>]
   [-repos_pwd <repos password>]
   ```

   Where `-dd` is the Connector Deployment Descriptor File (full path) and `-repos_pwd` is the Enterprise Manager Root (SYSMAN) Password.

   Table 2–5 lists the `emctl` parameters and provides a description for each.

Table 2–5 emctl Parameters

Parameter	Description
`connectorType.xml`	The connector deployment descriptor.
`ticketTemplate.xsl`	Fully qualified name of the ticket template file. The file resides in the Connector home directory: `$OMS_HOME/sysman/connector/Remedy_Connector` Oracle recommends that you use intuitive names since there might be notification methods created with the same names, and you have to choose one of them when you use the Auto Ticketing feature. Use `xsl` as the file extension since the format is XSLT. For example, `Remedy_DefaultCategory_LowPriority.xsl`. If the file is in a different directory, provide the complete path for the file.
`server`	Host name of the Enterprise Manager repository.
`port`	Listener port of the repository.
`database_sid/ Service Name for RAC DB`	Repository database instance ID or service name if you are using a RAC database as the repository.
`username`	Specify `SYSMAN`.
`password`	Password for `SYSMAN`.
`connectorTypeName`	Connector type name you define in `connectorType.xml`. For example, `"My Ticketing Connector"`. The double quotes ("") are mandatory.

Table 2–5 (Cont.) emctl Parameters

Parameter	Description
connectorName	Connector name. This should be the same as the connector type name. For example, " My Ticketing Connector"
templateName	An intuitive name for the ticket template that will be displayed in Enterprise Manager.
description	A short description for the ticket template. This description is also displayed in Enterprise Manager.
jar	Jar file, full jar file path
cname	Connector name
ctname	Connector type name
dd	Connector descriptor file, full path
repos_pwd	Enterprise Manager root (SYSMAN) password
t	Template file, full path
iname	Internal name (For details about this parameter, you can refer to the existing Connector documentation.)
tname	Template name
ttype	Template type
d	Template description

4. Log into the Enterprise Manager console and configure the event connector as described in the next section.
5. To register a template, run the following command:

```
emctl register_template connector [-t <template.xsl>]
[-repos_pwd <repos password>] [-ctname <connectorTypeName>]
[-cname <connectorName>] [-iname <internalName>] [-tname
<templateName>] [-ttype <templateType>] [-d <description>]
```

Where the following parameters apply:

```
-t              Template(full path)
-repos_pwd      Enterprise Manager Root (SYSMAN) Password
-ctname         Connector Type Name
-cname          Connector Name
-iname          Template Internal Name
-tname          Template Name Displayed
-ttype          Template Type
                <templateType> 1 - inbound transformation
                <templateType> 2 - outbound transformation
                <templateType> 3 - XML outbound transformation
-d              Description
```

Configuring the Event Connector

To configure the event connector:

1. As Super Administrator from the Enterprise Manager Grid Control console, click **Setup.**

   The Overview of Setup page appears.
2. Click **Management Connectors.**

The Management Connectors Setup page appears. A row with the connector name you defined should appear in this page.

3. Click the **Configure** icon of the connector that you registered.

   The connector configuration page and the General tab of the Configure Management Connector page appear.

4. Configure all mandatory fields indicated with an asterisk (*).

5. Click **OK.**

   The Management Connector Setup page reappears. The row for the connector you defined should have a check mark in the Configured column.

Managing Target Instances

When you deploy or install an event connector, the default target instance is created when the connector is configured instead of registered. The default target instance holds all the external events retrieved by this event connector if no other targets are created. You can optionally create specific targets to hold external events.

Creating Additional Target Instances

Do the following to create additional target instances:

1. From the Enterprise Manager console, click **Setup.**

   The setup links appear in the left margin.

2. Click **Management Connectors.**

   The Management Connectors home page appears. A row with the connector name you defined should appear in this page.

3. Click **Configure** icon for the Event Connector, then click the **Targets** tab.

   The Targets page appears.

4. In the Targets page, you can do the following:

   - To add a target, specify the Target Name and other required properties, then click **OK.**
   - To remove any added target, select the target, click **Remove,** then click **OK.**
   - To enable a generic target (not available or deleted), click **Enable.**

   After the transformation, if the incoming alert has a target name that matches the target name defined in Enterprise Manager, the alert is associated with that target. Otherwise, the alerts are redirected to the default target instance, `generic_target_type`. For example, if the connector contains a target type called `my_managed_device,` the default target instance will be `generic_my_managed_device`.

Sending Alerts to External Systems

Beginning with Enterprise Manager 10g Release 4, you can send alerts to external systems in real time through Web services. You can choose which type of alerts to send, such as critical alerts on a production database. Enterprise Manager 10g Release 4 supports metric alerts. You can also enforce the registration for external systems to retrieve real-time information.

To send alerts, your external system needs to support the following:

- "create event" and "update event" literal-document style Web services, as described in the following section.
- IP-based or SOAP header-based authentication.

You also need to change the connector built for alert exchange as described in the following sections.

Providing Deployment Descriptor Mappings

The following mappings are required:

- `createEvent`
- `updateEvent`

Deployment Descriptor File Example

The code in bold shows the additions for `createEvent` and `updateEvent`.

```
<ManagementConnector xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     xmlns="http://xmlns.oracle.com/sysman/connector">
  <Name>specify the connector name</Name>
  <Version>specify the 5-digit version, e.g., 1.0.0.0.0</Version>
  <Description>specify a short description</Description>
  <Category>EventConnector</Category>
<!-- Category is fixed to EventConnector. -->
  <EventConnector>
    <!-- If IsNewTargetType is set to true, it means the developers want to define
a new target type to hold external alerts. In that case, targetType.xml and
defaultTargetType.xml need to be shipped in the connector jar. -->
    <IsNewTargetType>true</IsNewTargetType>
    <EventService>
      <Method>getNewAlerts</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>getUpdatedAlerts</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
<Method>acknowledgeAlerts</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
```

```
    <EventService>
      <Method>setup</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>destroy</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>createEvent</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
    <EventService>
      <Method>updateEvent</Method>
      <WebServiceEndpoint>
        <![CDATA[specify the URL]]>
      </WebServiceEndpoint>
      <SOAPAction>specify the SOAP action</SOAPAction>
      <Namespace>specify the namespace</Namespace>
      <NamespacePrefix>specify the namespace prefix</NamespacePrefix>
    </EventService>
   <!-- BaseURL element is for the default UI for event connectors, the following
value should not be changed. -->
    <BaseURL>/em/console/connector/event/configuration</BaseURL>
  </EventConnector>
</ManagementConnector>
```

Providing Request Transformations

The following request transformation files are required:

- `createEvent_request.xsl`
- `updateEvent_request.xsl`

You can use the `EMEvent.xsd` schema to decide which Enterprise Manager fields you want to make available for mapping when sending alerts to external systems. The schema is available in the following path:

`$ORACLE_HOME/sysman/connector/common/schema/EMEvent.xsd`

Example files and the schema are shown below.

createEvent_request.xsl Transformation File Example

```
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
          xmlns:ns0="..."
          xmlns:a="http://xmlns.oracle.com/sysman/connector">
<xsl:template match="a:EMEvent">
  <ns0:InsertAlerts>
     <ns0:registrationId>
        <xsl:for-each select="a:Property">
            <xsl:if test="a:Name = 'REGISTRATION_ID'">
               <xsl:value-of select="a:Value"/>
                  </xsl:if>
                  </xsl:for-each>
     </ns0:registrationId>
        <ns0:alerts>
            <ns0:AlertInsert>
               <ns0:Name>
               <xsl:value-of select="a:MetricColumn"/>
               </ns0:Name>
               <ns0:ComputerName>
               <xsl:value-of select="a:TargetHost"/>
               </ns0:ComputerName>
               <ns0:ComputerDomain></ns0:ComputerDomain>
               <ns0:Description>
                   <xsl:value-of select="a:CollectionTime"/>
                   Received alert reported by Oracle Enterprise Manager:
                   Target Type: <xsl:value-of select="a:TargetType"/>
                   Target Name: <xsl:value-of select="a:TargetName"/>
                   Metric Name: <xsl:value-of select="a:MetricName"/>
                   Metric Column: <xsl:value-of select="a:MetricColumn"/>
                   <xsl:choose>
                   <xsl:when test="a:KeyValues">
                   Key Values: <xsl:for-each select="a:KeyValues">
                       <xsl:value-of select="."/>;
                       </xsl:for-each>
                   </xsl:when>
                   </xsl:choose>
                   Severity: <xsl:value-of select="a:Severity"/>
                   <xsl:choose>
                   <xsl:when test="normalize-space(a:NotificationRuleName)
                     != ''">
                   Notification Rule: <xsl:value-of
                     select="a:NotificationRuleName"/>
                   </xsl:when>
                   </xsl:choose>
                   Message: <xsl:value-of select="a:Message"/>
               </ns0:Description>
              <ns0:Severity>
                <xsl:choose>
                   <xsl:when test="a:SeverityCode = '15'">Success</xsl:when>
                   <xsl:when test="a:SeverityCode = '18'">Information</xsl:when>
                   <xsl:when test="a:SeverityCode = '20'">Warning</xsl:when>
                   <xsl:when test="a:SeverityCode =
                                              '25'">CriticalError</xsl:when>
                   <xsl:when test="a:SeverityCode = '115'">Success</xsl:when>
                   <xsl:when test="a:SeverityCode = '125'">Warning</xsl:when>
                   <xsl:when test="a:SeverityCode = '215'">Success</xsl:when>
                   <xsl:when test="a:SeverityCode = '225'">Warning</xsl:when>
                   <xsl:when test="a:SeverityCode = '315'">Success</xsl:when>
                   <xsl:when test="a:SeverityCode = '325'">Error</xsl:when>
                   <xsl:otherwise>Error</xsl:otherwise>
                </xsl:choose>
              </ns0:Severity>
```

```
      <ns0:Source>
      <xsl:value-of select="a:TargetType"/>: <xsl:value-of select="a:TargetName"/>
      </ns0:Source>
      <ns0:TimeRaised>
      <xsl:value-of select="a:CollectionTime"/>
      </ns0:TimeRaised>
                        </ns0:AlertInsert>
                  </ns0:alerts>
            </ns0:InsertAlerts>
      </xsl:template>
</xsl:stylesheet>
```

updateEvent_request.xsl Transformation File Example

```
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:ns0="..."
                xmlns:a="http://xmlns.oracle.com/sysman/connector">
      <xsl:template match="a:EMEvent">
            <ns0:UpdateAlerts>
                  <ns0:registrationId>
                        <xsl:for-each select="a:Property">
                              <xsl:if test="a:Name = 'REGISTRATION_ID'">
                                    <xsl:value-of select="a:Value"/>
                              </xsl:if>
                        </xsl:for-each>
                  </ns0:registrationId>
                  <ns0:updatedAlerts>
                        <ns0:AlertUpdate>
                              <ns0:AlertId>
                                    <xsl:value-of select="a:ExternalEventId"/>
                              </ns0:AlertId>

<ns0:OwnerNameUseExisting>true</ns0:OwnerNameUseExisting>

<ns0:SeverityUseExisting>false</ns0:SeverityUseExisting>
      <xsl:variable name="Severity">
           <xsl:choose>
               <xsl:when test="a:SeverityCode = '15'">Success</xsl:when>
               <xsl:when test="a:SeverityCode = '18'">Information</xsl:when>
               <xsl:when test="a:SeverityCode = '20'">Warning</xsl:when>
               <xsl:when test="a:SeverityCode = '25'">CriticalError</xsl:when>
               <xsl:when test="a:SeverityCode = '115'">Success</xsl:when>
               <xsl:when test="a:SeverityCode = '125'">Warning</xsl:when>
               <xsl:when test="a:SeverityCode = '215'">Success</xsl:when>
               <xsl:when test="a:SeverityCode = '225'">Warning</xsl:when>
               <xsl:when test="a:SeverityCode = '315'">Success</xsl:when>
               <xsl:when test="a:SeverityCode = '325'">Error</xsl:when>
               <xsl:otherwise>Error</xsl:otherwise>
           </xsl:choose>
      </xsl:variable>
      <ns0:Severity>
          <xsl:value-of select="$Severity"/>
      </ns0:Severity>
      <xsl:choose>
          <xsl:when test="a:SeverityCode = '15'">

<ns0:ResolutionStateUseExisting>false</ns0:ResolutionStateUseExisting>

    <ns0:ResolutionState>255</ns0:ResolutionState>
    </xsl:when>
```

```
    <xsl:otherwise>

<ns0:ResolutionStateUseExisting>true</ns0:ResolutionStateUseExisting>

    <ns0:ResolutionState>0</ns0:ResolutionState>
    </xsl:otherwise>
    </xsl:choose>

<ns0:LastModifiedByUseExisting>true</ns0:LastModifiedByUseExisting>
                        </ns0:AlertUpdate>
                    </ns0:updatedAlerts>
            </ns0:UpdateAlerts>
      </xsl:template>
</xsl:stylesheet>
```

EMEvent.xsd Schema

```
<?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="EMEventType.xsd"/>
  <xsd:element name= "EMEvent" type="EMEventType"/>
</xsd:schema>
```

EMEventType.xsd Schema

```
<?xml version="1.0" encoding="US-ASCII" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
     <xsd:complexType name="EMEventType">
          <xsd:sequence>
               <xsd:element name="TicketId" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="ConnectorId" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="EventId" type="EventIdType" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="EventGuid" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="ExternalEventId" type="xsd:string"
                            minOccurs="0" maxOccurs="1"/>
               <xsd:element name="ViolationId" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="EventType" type="EventTypeType" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="TargetType" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="TargetName" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="TargetProperties" type="TargetPropertiesType"
                            minOccurs="0" maxOccurs="unbounded"/>
               <xsd:element name="MetricColumn" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="MetricName" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="KeyColumn" type="xsd:string" minOccurs="0"
                            maxOccurs="1"/>
               <xsd:element name="KeyValues" type="xsd:string" minOccurs="0"
                            maxOccurs="unbounded"/>
```

```
                    <xsd:element name="Message" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="Severity" type="SeverityType" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="SeverityCode" type="SeverityCodeType"
                                 minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="JobID" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="JobName" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="JobType" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="JobOwner" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="JobStatus" type="JobStatusType" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="JobStatusCode" type="JobStatusCodeType"
minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="JobTarget" type="JobTargetType" minOccurs="0"
                                 maxOccurs="unbounded"/>
                    <xsd:element name="StepOutput" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="CollectionTime" type="xsd:dateTime"
                                 minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="EventPageURL" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="EMUser" type="xsd:string" minOccurs="1"
                                 maxOccurs="1"/>
                    <xsd:element name="HDUser" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="NotificationRuleName" type="xsd:string"
                                 minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="TargetHost" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="GracePeriodCheckMade" type="xsd:string"
                                 minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="TargetTimezone" type="xsd:string"
                                 minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="Property" type="PropertyType" minOccurs="0"
                                 maxOccurs="unbounded"/>
               </xsd:sequence>
          </xsd:complexType>
          <xsd:complexType name="TargetPropertiesType">
               <xsd:sequence>
                    <xsd:element name="name" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
                    <xsd:element name="value" type="xsd:string"
                                 minOccurs="0" maxOccurs="1"/>
               </xsd:sequence>
          </xsd:complexType>
          <xsd:complexType name="EventIdType">
               <xsd:sequence>
                    <xsd:element name="TargetId" type="xsd:string" minOccurs="1"
                                 maxOccurs="1"/>
                    <xsd:element name="MetricId" type="xsd:string" minOccurs="1"
                                 maxOccurs="1"/>
                    <xsd:element name="KeyId" type="xsd:string" minOccurs="0"
                                 maxOccurs="1"/>
               </xsd:sequence>
```

```
</xsd:complexType>
<xsd:simpleType name="SeverityType">
     <xsd:restriction base="xsd:string">
          <xsd:enumeration value="Clear"/>
          <xsd:enumeration value="Info"/>
          <xsd:enumeration value="Warning"/>
          <xsd:enumeration value="Critical"/>
          <xsd:enumeration value="Unreachable Clear"/>
          <xsd:enumeration value="Unreachable Start"/>
          <xsd:enumeration value="Blackout End"/>
          <xsd:enumeration value="Blackout Start"/>
          <xsd:enumeration value="Metric Error End"/>
          <xsd:enumeration value="Metric Error Start"/>
     </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="SeverityCodeType">
     <xsd:restriction base="xsd:string">
          <xsd:enumeration value="15"/>
          <xsd:enumeration value="18"/>
          <xsd:enumeration value="20"/>
          <xsd:enumeration value="25"/>
          <xsd:enumeration value="115"/>
          <xsd:enumeration value="125"/>
          <xsd:enumeration value="215"/>
          <xsd:enumeration value="225"/>
          <xsd:enumeration value="315"/>
          <xsd:enumeration value="325"/>
     </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="PropertyType">
     <xsd:sequence>
          <xsd:element name="Name" type="xsd:string"/>
          <xsd:element name="Value" type="xsd:string" nillable="true"/>
     </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="EventTypeType">
     <xsd:restriction base="xsd:string">
          <xsd:enumeration value="Alert"/>
          <xsd:enumeration value="JobStatus"/>
     </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="JobTargetType">
     <xsd:sequence>
          <xsd:element name="TargetName" type="xsd:string"/>
          <xsd:element name="TargetType" type="xsd:string"/>
     </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="JobStatusType">
     <xsd:restriction base="xsd:string">
          <xsd:enumeration value="Scheduled"/>
          <xsd:enumeration value="Running"/>
          <xsd:enumeration value="Error"/>
          <xsd:enumeration value="Failed"/>
          <xsd:enumeration value="Succeeded"/>
          <xsd:enumeration value="Suspended by User"/>
          <xsd:enumeration value="Suspended on Agent Unreachable"/>
          <xsd:enumeration value="Stopped"/>
          <xsd:enumeration value="Suspended on Lock"/>
          <xsd:enumeration value="Suspended on Event"/>
          <xsd:enumeration value="Suspended on Blackout"/>
```

```
                <xsd:enumeration value="Stop Pending"/>
                <xsd:enumeration value="Suspend Pending"/>
                <xsd:enumeration value="Inactive"/>
                <xsd:enumeration value="Queued"/>
                <xsd:enumeration value="Failed and Retried"/>
                <xsd:enumeration value="Waiting"/>
                <xsd:enumeration value="Skipped"/>
           </xsd:restriction>
     </xsd:simpleType>
     <xsd:simpleType name="JobStatusCodeType">
           <xsd:restriction base="xsd:string">
                <xsd:enumeration value="1"/>
                <xsd:enumeration value="2"/>
                <xsd:enumeration value="3"/>
                <xsd:enumeration value="4"/>
                <xsd:enumeration value="5"/>
                <xsd:enumeration value="6"/>
                <xsd:enumeration value="7"/>
                <xsd:enumeration value="8"/>
                <xsd:enumeration value="9"/>
                <xsd:enumeration value="10"/>
                <xsd:enumeration value="11"/>
                <xsd:enumeration value="12"/>
                <xsd:enumeration value="13"/>
                <xsd:enumeration value="14"/>
                <xsd:enumeration value="15"/>
                <xsd:enumeration value="16"/>
                <xsd:enumeration value="17"/>
                <xsd:enumeration value="18"/>
           </xsd:restriction>
     </xsd:simpleType>
</xsd:schema>
```

Providing Response Transformations

The following response transformation files are required:

- `createEvent_response.xsl`
- `updateEvent_response.xsl`

You can use the `EMEventResponse.xsd` schema to decide how to transform the response from external systems to the expected response from Enterprise Manager. The schema is available in the following path:

`$ORACLE_HOME/sysman/connector/common/schema/EMEventResponse.xsd`

After the transformation, Enterprise Manager expects a `<SuccessFlag>` and one of the following elements depending on the value:

- **true** — Enterprise Manager expects an `<externalEventID>`, which is the ID of the alert in the external system. This is used to update the alert in the future.
- **false** — Enterprise Manager expects an `<ErrorMessage>`. Enterprise Manager annotates the alert with the `externalEventID` or `ErrorMessage`.

Example files and the schema are shown below.

createEvent_response.xsl Response Transformation File Example

```
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:ns0="..."
```

```
               xmlns:a="http://xmlns.oracle.com/sysman/connector">
 <xsl:template match="ns0:InsertAlertsResult">
  <a:EMEventResponse>
   <xsl:choose>
    <xsl:when test="ns0:InsertedAlerts/ns0:SuccessfulAlertInsert/ns0:NewAlertId">
     <a:SuccessFlag>true</a:SuccessFlag>
     <a:ExternalEventId>
      <xsl:value-of
select="ns0:InsertedAlerts/ns0:SuccessfulAlertInsert/ns0:NewAlertId"/>
     </a:ExternalEventId>
    </xsl:when>
    <xsl:otherwise>
     <a:SuccessFlag>false</a:SuccessFlag>
     <a:ErrorMessage>
      <xsl:value-of select="ns0:FailedAlerts/ns0:FailedAlertInsert/ns0:Error"/>
     </a:ErrorMessage>
    </xsl:otherwise>
   </xsl:choose>
  </a:EMEventResponse>
 </xsl:template>
</xsl:stylesheet>
```

updateEvent_response.xsl Response Transformation File Example

```
<?xml version='1.0' ?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
               xmlns:ns0="..."
               xmlns:a="http://xmlns.oracle.com/sysman/connector">
 <xsl:template match="ns0:UpdateAlertsResult">
  <a:EMEventResponse>
   <xsl:choose>
    <xsl:when test="ns0:UpdatedAlerts/ns0:guid">
     <a:SuccessFlag>true</a:SuccessFlag>
     <a:ExternalEventId>
      <xsl:value-of select="ns0:UpdatedAlerts/ns0:guid"/>
     </a:ExternalEventId>
    </xsl:when>
    <xsl:otherwise>
     <a:SuccessFlag>false</a:SuccessFlag>
     <a:ErrorMessage>
      <xsl:value-of
select="ns0:FailedUpdatedAlerts/ns0:FailedAlertUpdate/ns0:Error"/>
     </a:ErrorMessage>
    </xsl:otherwise>
   </xsl:choose>
  </a:EMEventResponse>
 </xsl:template>
</xsl:stylesheet>
```

EMEventResponse.xsd Schema

```
<?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="EMEventResponse">
          <xsd:complexType>
               <xsd:sequence>
                    <xsd:element name="SuccessFlag" type="xsd:boolean"
                                 minOccurs="1" maxOccurs="1"/>
```

```
                    <xsd:choice>
                         <xsd:element name="ExternalEventId" type="xsd:string"
                                      minOccurs="1" maxOccurs="1"/>
                         <xsd:element name="ErrorMessage" type="xsd:string"
                                      minOccurs="1" maxOccurs="1"/>
                    </xsd:choice>
               </xsd:sequence>
          </xsd:complexType>
     </xsd:element>
</xsd:schema>
```

Enabling an SSL Connection for HTTPS

Follow the steps provided in this section if you choose HTTPS as the protocol to establish a connection between the external eventing system and Enterprise Manager.

Generating Certificate Request File

Generate a certificate request file for the external eventing system and send it to the Certificate authority, such as VeriSign.

> **Note:** The certificate request file is dependent on the Web server the external eventing system uses.

Importing the Certificate from the Certificate Authority

After you get the certificate, import it to the Web server the external eventing system uses. The import mechanism varies depending on the Web server the external eventing system uses.

Adding Signed Certificates to Wallet Manager

> **Note:** Oracle Wallet Manager is available at `$ORACLE_HOME/bin` on OMS. See the *Oracle Application Server Administrator's Guide* for details.

1. Get the port number from *user_projects/domains/EMGC_DOMAIN/config/config.xml*:

```
<server>
<name>EMGC_ADMINSERVER</name>
<ssl>
<name>EMGC_ADMINSERVER</name>
<enabled>true</enabled>
<hostname-verification-ignored>true</hostname-verification-ignored>
<listen-port>7022</listen-port>
</ssl>
<listen-port-enabled>false</listen-port-enabled>
<listen-address>server_name</listen-address>
</server>
```

2. Connect to the WebLogic Admin Console:
   a. Connect to *https://server_name:7022/console/*, login: [username]/[password]
   b. Navigate to *Environment->Servers->EMGC_DOMAIN->Keystore tab*

c. If two-way SSL is required for external web service, go to *SSL->Advanced->Use Server Certs* and be sure it is checked

d. Make sure that *CA signed the external system server's certificate* is added to trustStore specified in the Keystore tab, for example, *WebLogic's DemoTrust*, or *java standard cacerts*

See Also: For information on creating an Oracle Wallet, see "Creating and Viewing Oracle Wallets with orapki" in the *Oracle Database Advanced Security Administrator's Guide, 10g Release 2 (10.2).*

3

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:

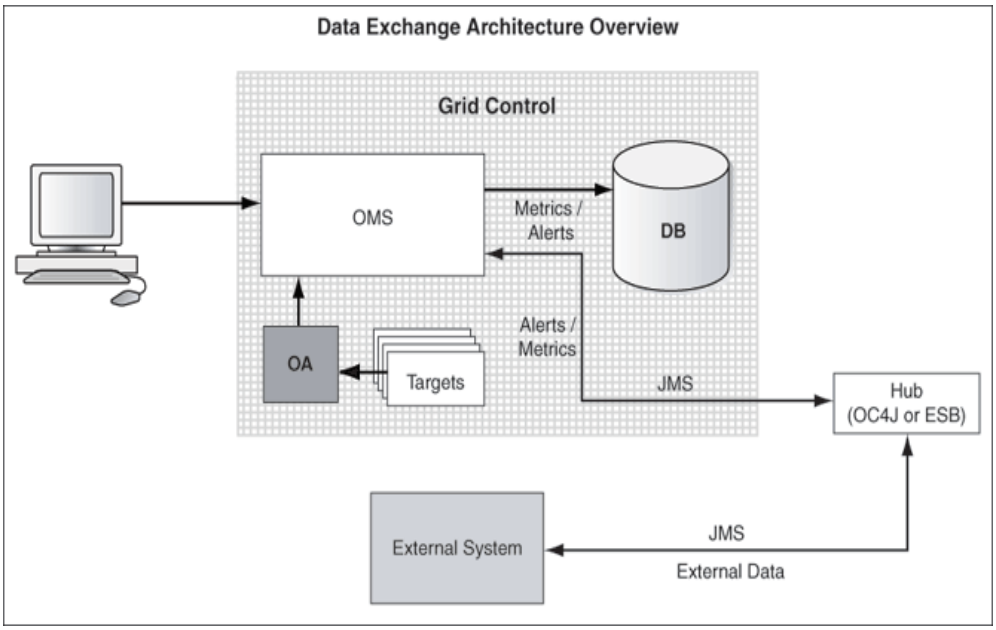
Introduction

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 OBAM 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.

Figure 3–1 Data Exchange Architecture



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 3–1 Data Exchange between Enterprise Manager and External Management System

Data Exchange	Requirement	Purpose
From Enterprise Manager to external management system	▪ Metric Data	▪ 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.
	▪ Alert Data	▪ Better reporting, event notification, and corrective actions.
From external management system to Enterprise Manager	▪ Business Indicators	▪ Better reporting and topology analysis.
	▪ Business Events	▪ Generate a comprehensive SLA in the Enterprise Manager environment.

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

Data Forwarding Frequency Options and Modes

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

- Only real-time metric and availability values are forwarded. Historical data is not forwarded.

- 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.

  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.
- Service-level Agreement (SLA) data sent out is the SLA value for the past 24 hours. If an outbound session with an SLA metric is scheduled at January 15th at 4:00 p.m., the value sent out is the SLA value from January 14th at 4:01 pm to January 15th at 4:00 p.m.

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.

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 Grid 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" on page 3-5

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 3-24

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

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 Grid 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.

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 3-24

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.

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.

Setting up a Data Exchange Connector

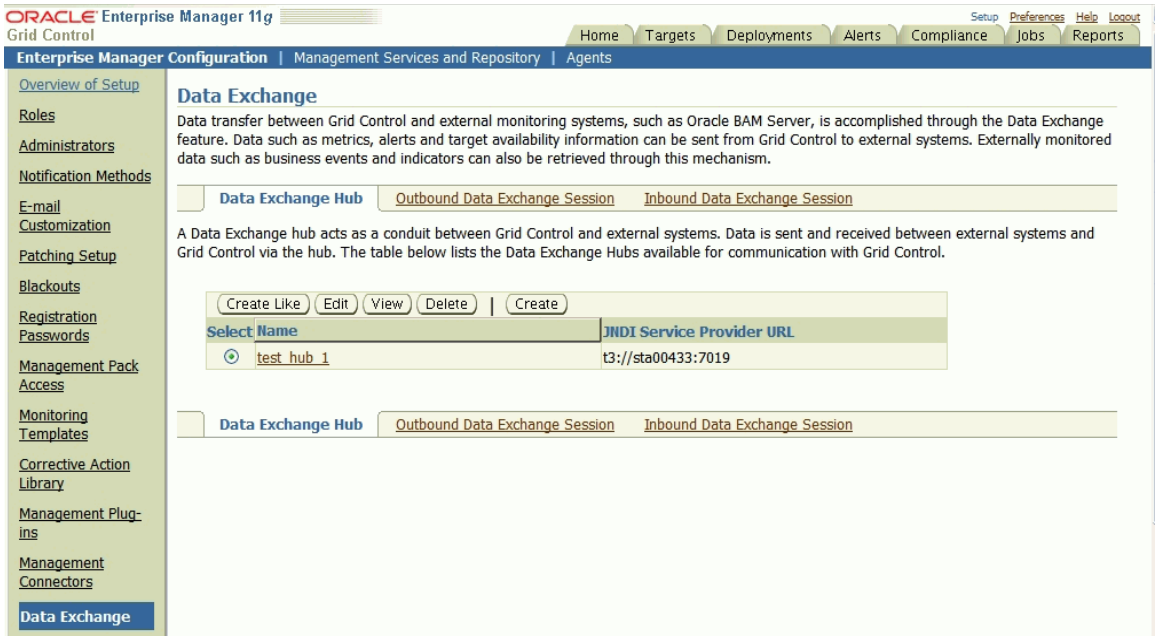
1. From Enterprise Manager Grid Control, click **Setup.**

   The setup links appear in the left margin.

2. Click **Data Exchange.**

   The Data Exchange page appears as shown in Figure 3–2.

Figure 3–2 Data Exchange Page



3. 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

Creating a Data Exchange Hub

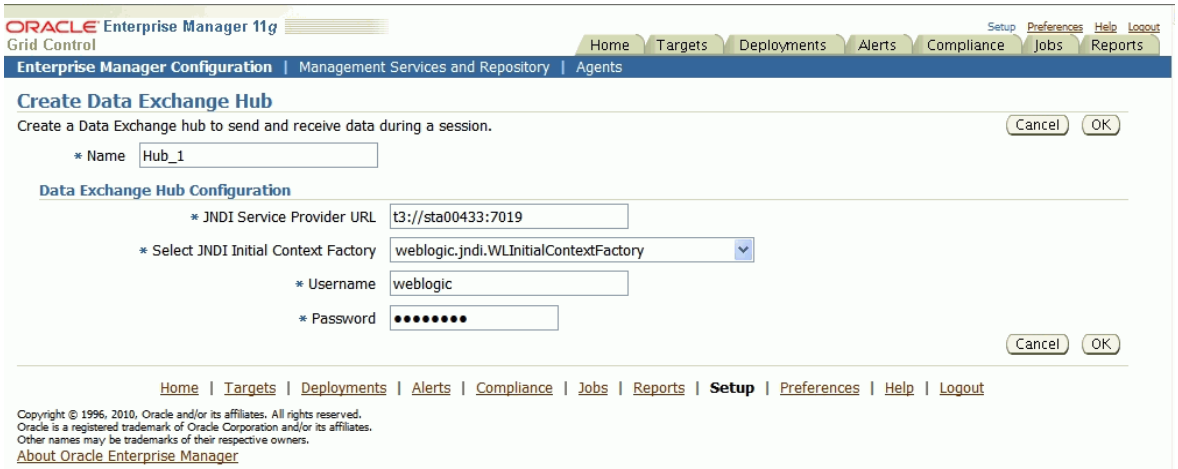
Any certified JMS-compliant server can act as a hub. The following are tested and certified:

- 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 (Figure 3–2), click **Create**. The Create Data Exchange Hub page appears, as shown in Figure 3–3.

Figure 3–3 Create Data Exchange Hub Page



a. Specify a unique name for the Data Exchange hub.

b. Provide the JNDI Service Provider URL for the Data Exchange hub.

c. Select a context factory name from the list according to the matrix in Table 3–2.

Table 3–2 Context Factory Name

Hub Corresponds to ...	Context Factory
WebLogic Server	weblogic.jndi.WLInitialContextFactory
10.1.2.X OC4J	com.evermind.server.rmi.RMIInitialContextFactory
10.1.3.X OC4J	oracle.j2ee.rmi.RMIInitialContextFactory
Non-WebLogic Server and non-OC4j JMS Server	For third-party servers, select Other and provide a factory name in the Enter JNDI Initial Context Factory Name field.

d. Provide the user credentials to access this hub.

2. Configure the JMS server with the required JMS topic names and/or queue names.

> **See Also:** "Inbound JMS Destinations" on page 27

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.

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" on page 3-5.

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 a BAM user or a comma-separated list of 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 Grid 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" on page 3-3.

g. 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**.

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 createEMSYSJMSSystemResource.py <jndi provider URL>
   <username> <password> <WLS server name>
   ```

   Example:

```
java weblogic.WLST createEMSYSJMSSystemResource.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
- **Queues**:
  - jms/EMSYSTargetsQueue
  - jms/EMSYSMetricsQueue
  - jms/EMSYSAlertsDataQueue
  - jms/EMSYSMetricsDataQueue
  - jms/EMSYSSecurityFilterQueue
  - jms/EMSYSTargetStatusQueue
  - jms/EMSYSTargetSLAQueue

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

   ```
   java weblogic.WLST deleteEMSYSJMSSystemResource.py <jndi provider URL>
   <username> <password>
   ```

   Example:

   ```
   java weblogic.WLST deleteEMSYSJMSSystemResource.py "t3://localhost:7001"
   weblogic welcome1
   ```

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

Table 3–3 JMS Destination for Targets

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSTargetsTopic or jms/EMSYSTargetsQueue
JMS Message Type	Text message
Description	Target metadata information, such as target name and type are sent on this destination.

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

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSMetricsTopic or jms/EMSYSMetricsQueue
JMS Message Type	Text message
Description	Metric metadata information, such as metric name, column, and target type are sent on this destination.

Table 3–5 JMS Destination for Metric Data

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSMetricsDataTopic or jms/EMSYSMetricsDataQueue
JMS Message Type	Text message
Description	This destination is used to send the actual metric values.

Table 3–6 JMS Destination for Target Status

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSTargetStatusTopic or ms/EMSYSTargetStatusQueue

Table 3–6 (Cont.) JMS Destination for Target Status

Details	Description
JMS Message Type	Text message
Description	Target status information is sent on this destination.

Table 3–7 JMS Destination for Security Filter

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSSecurityFilterTopic
JMS Message Type	Text message
Description	Security filter information is sent on this destination.

Table 3–8 JMS Destination for Alert Data

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSAlertsDataTopic
JMS Message Type	Text message
Description	Alerts are sent on this destination.

Table 3–9 JMS Destination for SLA Data

Details	Description
ConnectionFactory Name	jms/EMSYSTopicConnectionFactory or jms/EMSYSQueueConnectionFactory
Destination Name	jms/EMSYSATargetSLATopic or jms/EMSYSATargetSLAQueue
JMS Message Type	Text message
Description	The target SLA is sent on this destination. SLA metrics are only shown for Service targets.

Outbound Message Schema

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

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 3–10 Normalized Target Message

Title	Description
Path Expression	EMSYSData/Target
Schema File Location	$ORACLE_HOME/sysman/bam/OutboundTarget.xsd
Destination	jms/EMSYSTargetsTopic or jms/EMSYSTargetsQueue

Schema	`<?xml version="1.0" encoding="UTF-8"?>`
	`<xs:schema`
	`targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExch`
	`ange/10203/OutboundData/"`
	`xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10`
	`203/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="Target" type="de:TargetType"`
	`minOccurs="0" maxOccurs="unbounded"/>`
	`</xs:sequence>`
	`</xs:complexType>`
	`<!-- Define the Target Type -->`
	`<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:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10`
	`203/OutboundData/">`
	`<Target>`
	
	`<TargetName>/ade/foo_core9/oracle.stacd16.us.oracle.com_home</Targe`
	`tName>`
	`<TargetType>oc4j</TargetType>`
	`<TargetGUID>852464ac3e4176460458297faaffe926</TargetGUID>`
	`</Target>`
	`<Target>`
	`<TargetName>stacd16.us.oracle.com</TargetName>`
	`<TargetType>host</TargetType>`
	`<TargetGUID>00645a665bfd9b72b2a6bb6ef49606b0</TargetGUID>`
	`</Target>`
	`</de:EMSYSData>`

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 3–11 Normalized Metric Message

Title	Description
Path Expression	EMSYSData/Metric
Schema File Location	$ORACLE_HOME/sysman/bam/OutboundSecurityFilter.xsd
Destination	jms/EMSYSMetricsTopic or jms/EMSYSMetricsQueue
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/" 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="Metric" type="de:MetricType" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> </xs:complexType> <!-- 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"/> </xs:all> </xs:complexType> </xs:schema>
Sample	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"> <Metric> <MetricColumn>DiskActivityavserv</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 3–12 Normalized Security Filter Message

Title	Description
Path Expression	EMSYSData/SecurityFilter
Schema File Location	$ORACLE_HOME/sysman/bam/OutboundMetric.xsd
Destination	jms/EMSYSSecurityFilterTopic or jms/EMSYSSecurityFilterQueue
Schema	<?xml version="1.0" encoding="UTF-8"?> <xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExcha nge/10203/OutboundData/" xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/102 03/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	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/102 03/OutboundData/"> <SecurityFilter> <SessionName>LoanSession</SessionName> <UserName>LoanAdminUser</UserName> </SecurityFilter> </de:EMSYSData>

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:

Table 3–13 Normalized Metric Data Message

Title	Description
Path Expression	EMSYSData/MetricData
Schema File Location	$ORACLE_HOME/sysman/bam/ OutboundNormalizedMetricsData.xsd
Destination	jms/EMSYSMetricsDataTopic or jms/EMSYSMetricsDataQueue

Table 3–13 (Cont.) Normalized Metric Data Message

Title	Description
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/" 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"/> <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	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"> <MetricData> <SessionName>Session1</SessionName> <TargetGUID>00645a665bfd9b72b2a6bb6ef49606b0</TargetGUID> <MetricGUID>df8067b7d515747434b58d69230b8451</MetricGUID> <Timestamp>2001-12-17T09:30:47-05:00</Timestamp> <Value>3.14159</Value> </MetricData> </EMSYSData>

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 3–14 Normalized Alert Message

Title	Description
Path Expression	EMSYSData/Alert
Schema File Location	$ORACLE_HOME/sysman/bam/ OutboundNormalizedAlertsData.xsd

Table 3–14 (Cont.) Normalized Alert Message

Title	Description
Destination	jms/EMSYSAlertsDataTopic or jms/EMSYSAlertsDataQueue
Schema	<?xml version="1.0" encoding="UTF-8"?> <xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/" xmlns:xs="http://www.w3.org/2001/XMLSchema" 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="TargetGUID" type="xs:string"/> <xs:element name="MetricGUID" type="xs:string"/> <xs:element name="Timestamp" type="xs:dateTime"/> <xs:element name="Severity" type="xs:string"/> <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	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"> <Alert> <MetricGUID>0c71a1afac2d7199013837da35522c08</MetricGUID> <Value>25.35</Value> <Message>CPU Utilization is 25.35%, crossed warning (15) or critical (95) threshold.</Message> <Severity>Warning</Severity> <Timestamp>07/13/2006 14:59:42</Timestamp> <SessionName>Session9</SessionName> <TargetGUID>00645a665bfd9b72b2a6bb6ef49606b0</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 3–15 Normalized Target Availability Message

Title	Description
Path Expression	EMSYSData/TargetStatus
Schema File Location	$ORACLE_HOME/sysman/bam/OutboundNormalizedTargetStatus.xsd
Destination	jms/EMSYSTargetStatusTopic or jms/EMSYSTargetStatusQueue

Schema	`<?xml version="1.0" encoding="UTF-8"?>`
	`<xs:schema`
	`targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchang`
	`e/10203/OutboundData/" xmlns:xs="http://www.w3.org/2001/XMLSchema"`
	`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>`

Sample	`<de:EMSYSData`
	`xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203`
	`/OutboundData/">`
	`<TargetStatus>`
	`<Status>1</Status>`
	`<Timestamp>07/11/2006 16:21:53</Timestamp>`
	`<SessionName>Session1</SessionName>`
	`<TargetGUID>00645a665bfd9b72b2a6bb6ef49606b0</TargetGUID>`
	`</TargetStatus>`
	`</de:EMSYSData>`

Possible status values are provided in the following table:

Value	Status
1	Target is up and reachable
-1	■ Target is down ■ Metric error ■ Agent is down
0	■ Blackout ■ Target is not monitored ■ Target is unknown

Normalized Target SLA Message

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

Table 3–16 Normalized Target SLA Message

Title	Description
Path Expression	EMSYSData/TargetSLA
Schema File Location	$ORACLE_HOME/sysman/bam/OutboundNormalizedTargetSLA.xsd
Destination	jms/EMSYSTargetSLATopic or jms/EMSYSTargetSLAQueue
Schema	<?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 3–16 (Cont.) Normalized Target SLA Message

Title	Description
Sample	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/ 10203/OutboundData/"> <TargetSLA> <SLA>100</SLA> <Timestamp>07/11/2008 16:21:53</Timestamp> <SessionName>Session1</SessionName> <TargetGUID>00645a665bfd9b72b2a6bb6ef49606b0</TargetGUID> </TargetSLA> </de:EMSYSData>

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:

Table 3–17 Denormalized Metric Data Message

Title	Description
Path Expression	EMSYSData/MetricData
Schema File Location	$ORACLE_HOME/sysman/bam/ OutboundDenormalizedMetricsData.xsd
Destination	jms/EMSYSMetricsDataTopic or jms/EMSYSMetricsDataQueue

Table 3–17 (Cont.) Denormalized Metric Data Message

Title	Description
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/" 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"/> <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"/> <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	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"> <MetricData> <SessionName>Session1</SessionName> <TargeName>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> </MetricData> </EMSYSData>

Denormalized Alert Message

Schema of a denormalized alert message is as follows:

Table 3–18 Denormalized Alert Message

Title	Description
Path Expression	EMSYSData/Alert

Table 3–18 (Cont.) Denormalized Alert Message

Title	Description
Schema File Location	$ORACLE_HOME/sysman/bam/ OutboundDenormalizedAlertsData.xsd
Destination	jms/EMSYSAlertsDataTopic or jms/EMSYSAlertsDataQueue
Schema	<?xml version="1.0" encoding="UTF-8"?> <xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/" xmlns:xs="http://www.w3.org/2001/XMLSchema" 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"/> <xs:element name="Severity" type="xs:string"/> <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	<de:EMSYSData xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/OutboundData/"> <Alert> <SessionName>Session9</SessionName> <TargeName>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>07/13/2006 14:59:42</Timestamp> </Alert> </de:EMSYSData>

Denormalized Target Availability Message

Schema of a denormalized target availability message is as follows:

Table 3–19 Denormalized Target Availability Message

Title	Description
Path Expression	EMSYSData/TargetStatus
Schema File Location	$ORACLE_HOME/sysman/bam/ OutboundDenormalizedTargetStatus.xsd
Destination	jms/EMSYSTargetStatusTopic or jms/EMSYSTargetStatusQueue
Schema	<?xml version="1.0" encoding="UTF-8"?> <xs:schema targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchang e/10203/OutboundData/" xmlns:xs="http://www.w3.org/2001/XMLSchema" 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:TargetStatusType" 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="TargetName" type="xs:string"/> <xs:element name="TargetType" type="xs:string"/> <xs:element name="Status" 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/"> <TargetStatus> <Status>1</Status> <Timestamp>07/11/2006 16:21:53</Timestamp> <SessionName>Session1</SessionName> <TargeName>OC4J 10.1.3</TargetName> <TargetType>generic_service</TargetType> </TargetStatus> </de:EMSYSData>

Denormalized Target SLA Message

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

Table 3–20 Denormalized Target SLA Message

Title	Description
Path Expression	EMSYSData/TargetSLA
Schema File Location	$ORACLE_HOME/sysman/bam/OutboundDenormalizedTargetSLA.xsd
Destination	jms/EMSYSTargetSLATopic or jms/EMSYSTargetSLAQueue
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>07/11/2006 16:21:53</Timestamp> <SessionName>Session1</SessionName> <TargeName>OC4J 10.1.3</TargetName> <TargetType>generic_service</TargetType> </TargetSLA> </de:EMSYSData>

Creating an Inbound Data Exchange Session

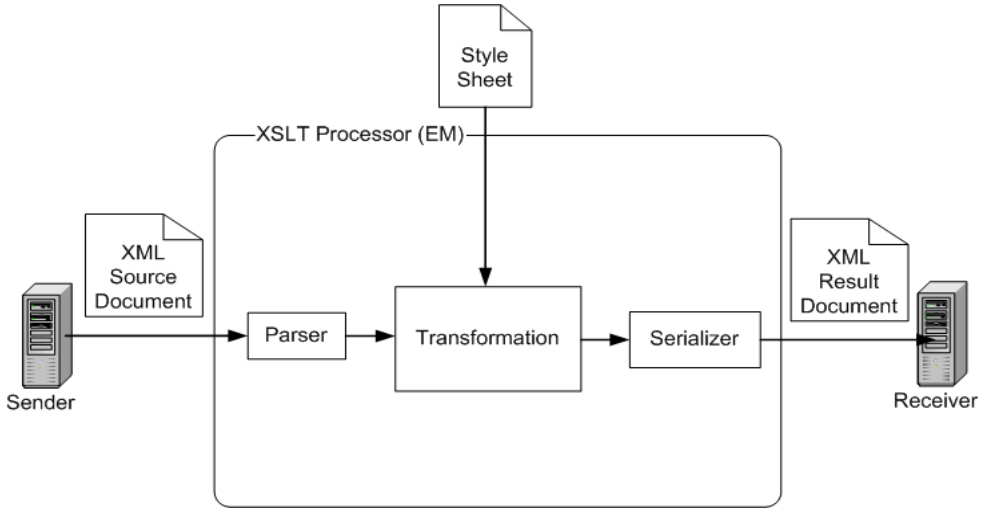
To create an inbound data exchange session, specify the input provided in the following procedure for the respective pages of the setup wizard.

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" on page 3-5.
   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 Grid 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 Grid 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 3–4. 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.

      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 Grid Control end rather than changing the message format itself.

Figure 3–4 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 Grid 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 Grid 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 Grid 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, shown in Figure 3–4. 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 Grid 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 Grid 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 Grid 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 interval at which you want the session to run, and whether it should be repeated indefinitely or until a specified time and date.

6. 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.

7. Click **Finish**. The Inbound 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**.

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.

Inbound Message Schemas

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

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

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

Unqualified XML Message Sample

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

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 3–1.
- 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 3–1.
- 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 (2006-10-30 17:43:19.474):

```
<po: PurchaseOrder xmlns:po:"http://acme.com/Orders">
  <OrderAmount>5000</OrderAmount>
  <NoOfItems> 15 </NoOfItems>
  <Timestamp>2006-10-30 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 3–1 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:

```
<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 datasource).
- If `TargetName` is not part of the message, it defaults to the target to which the datasource 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.

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 version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/Inbo
undEvents/"
xmlns:de="http://xmlns.oracle.com/EnterpriseManager/DataExchange/10203/InboundEven
ts/" xmlns:xs="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>
```

Integrating Enterprise Manager with OBAM

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

Supported Versions

The tested and certified versions of OBAM 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

Note: Oracle BAM Server 10gR2 and 10gR3 are OC4J-based. Oracle BAM Server 11gR1+ is WebLogic Web Server-based. Consequently, the setup steps differ considerably.

The following sections provide basic steps and guidelines. Refer to the Oracle BAM Server documentation for specific information and details.

Setting up the Data Flow from Enterprise Manager to OBAM

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

1. Import required artifacts, explained in Importing OBAM Artifacts for an Outbound Session.
2. Update JNDI details, explained in Updating JNDI.
3. Run the link plans shown in Table 3–23. (This is only needed for OBAM 10g R3 and previous versions.)

Importing OBAM Artifacts for an Outbound Session

OBAM server is not packaged or installed as part of Enterprise Manager. It is assumed that an OBAM 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 OBAM artifacts needed for the integration with OBAM server, as super user, run the following script:

- For Oracle BAM 11gR1 (11.1.1.1.0) or later versions:

```
ICommand cmd=import file=emsys_all_11.xml
```

- For Oracle BAM 10gR3 or older versions:

```
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 OBAM artifacts:

- EM-BAM Data Objects
- EM-BAM EMS Definitions
- EM-BAM Enterprise Link Plans (only when emsys_all_10.xml is used). See Table 3–23.

EM-BAM Data Objects

Table 3–21 lists the data objects the Import command creates.

Table 3–21 EM-BAM Data Objects

Data Object	Description
/SYSMAN/EMSYSTargets	Contains target metadata information, such as target name and target type.
/SYSMAN/EMSYSMetrics	Contains metric metadata, such as metric name, metric column, and target type.

Table 3–21 (Cont.) EM-BAM Data Objects

Data Object	Description
/SYSMAN/EMSYSAlertsData	Holds 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.
/SYSMAN/EMSYSTargetSLAData	Holds the target SLA information received from Enterprise Manager.
/SYSMAN/EMSYSTargets	Contains target metadata information, such as target name and target type.
/SYSMAN/EMSYSMetricsData	Holds the metrics information received from Enterprise Manager. It has information about the target and metric for which metrics are received along with metric value and timestamp.
/SYSMAN/EMSYSSecurityFilter	Acts as the security filter for all other data objects. It contains the session name and users who can access the corresponding session data.

EM-BAM EMS Definitions

Table 3–22 lists the enterprise message sources the Import command creates.

Table 3–22 EM-BAM EMS Definitions

Data Definition	Description
EMSYSMetricsEMS	Contains the EMS definition for incoming metric metadata listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsTopic.
EMSYSTargetsEMS	Contains the EMS definition for incoming target metadata listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetsTopic.
EMSYSSecurityFilterEMS	Contains the EMS definition for security filter data listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSSecurityFilterTopic.
EMSYSAlertsDataEMS	Contains the EMS definition for incoming alerts listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSAlertsDataTopic.
EMSYSMetricsDataEMS	Contains the EMS definition for incoming metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSMetricsDataTopic.
EMSYSTargetStatusDataEMS	Contains the EMS definition for incoming target status messages metrics listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetStatusTopic.
EMSYSTargetSLAEMS	Contains the EMS definition for incoming target SLA messages listening on jms/EMSYSTopicConnectionFactory and jms/EMSYSTargetSLATopic.
EMSYSMetricsEMS-Queue	Contains the EMS definition for incoming metric metadata listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsQueue.
EMSYSTargetsEMS-Queue	Contains the EMS definition for incoming target metadata listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetsQueue.

Table 3–22 (Cont.) EM-BAM EMS Definitions

Data Definition	Description
EMSYSSecurityFilterEMS-Queue	Contains the EMS definition for security filter data listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSSecurityFilterQueue.
EMSYSAlertsDataEMS-Queue	Contains the EMS definition for incoming alerts listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSAlertsDataQueue.
EMSYSMetricsDataEMS-Queue	Contains the EMS definition for incoming metrics listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSMetricsDataQueue.
EMSYSTargetStatusDataEMS-Queue	Contains the EMS definition for incoming target SLA messages listening on jms/EMSYSQueueConnectionFactory and jms/EMSYSTargetSLAQueue.

Updating JNDI

You should update all EMSs described in Table 3–22 to reflect the correct JNDI initial context factory and provider URLs of your JMS servers. To do this:

1. In the OBAM Console, click **Architect.**

   The BAM Architect screen appears (Figure 3–5).

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.

   - For 11gR1 or new versions of OBAM:

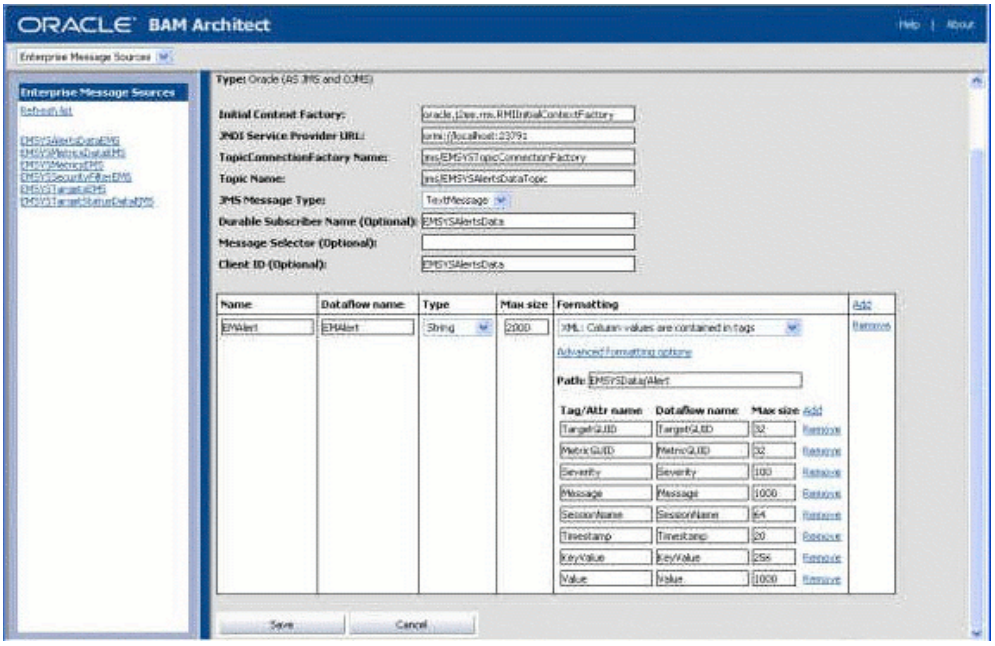
     Enable Global trust or Cross domain trust between the Data hub WebLogic server and the OBAM WebLogic server. Refer to WebLogic documentation for details.

   - For 10gR3 or older versions of OBAM:

     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 OBAM services.

Figure 3–5 OBAM Architect Screen



EM-BAM Enterprise Link Plans

Besides the BAM data objects (Table 3–21) and EMS definitions (Table 3–22), the link plans shown in Table 3–23 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 3–23 are created based on the Import command.

Table 3–23 EM-BAM Link Plans

Plan	Description
EMSYSMetricsPlan	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.
EMSYSTargetsPlan	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.
EMSYSSecurityFilterPlan	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.
EMSYSAlertsDataPlan	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.
EMSYSMetricsDataPlan	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.

Table 3–23 (Cont.) EM-BAM Link Plans

Plan	Description
EMSYSTargetStatusDataPlan	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.
EMSYSTargetSLADataPlan	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.
EMSYSAlertsDataRollup	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.
EMSYSMetricsDataRollup	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.
EMSYSTargetStatusDataRollup	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.

Additional EM-BAM Supplemental Data Objects

Table 3–24 lists the supplemental data objects related to repositories for archived data.

Table 3–24 EM-BAM Supplemental Data Objects

Plan	Description
/SYSMAN/EMSYSAlertsData.Archive	Holds the archived data for /SYSMAN/EMSYSAlertsData.
/SYSMAN/EMSYSMetricsData.Archive	Holds the archived data for /SYSMAN/EMSYSMetricsData.
/SYSMAN/EMSYSTargetStatusData.Archive	Holds the archived data for /SYSMAN/EMSYSTargetStatusData.

Setting up the Data Flow from OBAM to Enterprise Manager

Unlike the Enterprise Manager to OBAM server data transfer, where pre-defined OBAM 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 OBAM to Enterprise Manager. This is because the Data Object or EMS are unknown.

Consequently, for inbound data transfer from OBAM 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 OBAM 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 OBAM documentation for specific information.

End-to-End Flow

After you configure the OBAM 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 OBAM:

- OBAM 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 OBAM 10g or older versions) described in Table 3–23 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 OBAM server. OBAM 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.

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 Grid 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 Grid 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:

```
<topic-connection-factory/>
    location="jms/EMSYSTopicConnectionFactory"

<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"
    location="jms/EMSYSTargetSLAQueue"
    <description>Queue for target SLA</description>
</queue>

<queue>
    name="EMSYSTargetsQueue"
    location="jms/EMSYSTargetsQueue"
    <description>Queue for targets metadata</description>
</queue>
```

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.

Data Exchange Hub Connection Errors

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

- Grid Control log files
- Grid 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:

  ```
  2008-01-29 17:37:28.259 NOTIFICATION J2EE RMI-00004 Invalid
  username or password for default (oc4jadmin). Authentication
  failed.
  08/02/28 17:37:28 INFO: RMIProto  .readConnectionHeader Local
  ORMI version = 1. 3 different from remote version 1.1 will
  use 1.1
  2008-01-29 17:37:28.290 ERROR [RealmLoginModule]
  authentication failed
  ```

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 Grid Control as follows:
   a. Log in to Grid Control as a super user/administrator.
   b. From Enterprise Manager Grid Control, click **Setup.**

The setup links appear in the left margin.

c. Click **Data Exchange**.

The Data Exchange page appears (Figure 3–2).

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.

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_name*`NotifDevice`, 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_name*`NotifRule`, 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 Grid 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.

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 OBAM) 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.

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 Grid 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.

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.

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:
  http://www.oracle.com/technology/products/integration/bam/index.html
- Oracle Data Integrator:

  http://www.oracle.com/technology/products/oracle-data-integrator/index.html
- Java Message Service:

  http://java.sun.com/products/jms/
- Oracle Containers for J2EE (OC4J):

  http://www.oracle.com/technology/tech/java/oc4j/index.html
- Oracle Enterprise Service Bus (ESB):

  http://www.oracle.com/technology/products/integration/esb/index.html
- Enterprise Manager Metrics:

  Oracle Enterprise Manager Framework, Host, and Third-Party Metric Reference Manual available at the following URL:

http://www.oracle.com/technology/documentation/oem.html

4

Reference Tables

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.

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 4–1 setModel Request Elements

Path Type	Description	Data Type
<EMModel xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xmlns="**http://xmlns.oracle.com/sysman/connector/base/msi**">	EMModel element.	Complex Type
<RequestHeader/>	Request header. See Table 4–2.	Complex Type
<Credential>	Credential for Enterprise Manager.	Complex Type
<Name>sysman</Name>	User name.	String

Table 4–1 (Cont.) setModel Request Elements

Path Type	Description	Data Type
<Password>welcome1 </Password>	Password.	String
</Credential>	End of Credential.	Complex Type
<AggregateTarget>	EMModel 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.	Complex Type
<Name>CRS30</Name>	Name of the cluster.	String
<Type>cluster</Type>	Aggregate target type.	String (enumeration: "cluster" "rac_database"
<Target>	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.	Complex Type
<Name>bjx30</Name>	Name of the host.	String
<Type>host</Type>	Target type. In the cluster aggregate target, it should be set to "host."	String
<Host>bjx30</Host>	Name of the host.	String
<Credential>	Credential of the host.	Complex Type
<Name>oracle</Name>	User name.	String
<Password>welcome1 </Password>	Password.	String
</Credential>	End of Credential.	Complex Type
<Property/>	Property of the target. See the corresponding table in this section for the properties of the "host" target type.	Complex Type
</Target>	End of Target.	Complex Type
<Property/>	Property of the cluster aggregate target. See the corresponding table in this section for the properties of the "cluster" aggregate target type.	Complex Type
</AggregateTarget>	End of AggregateTarget.	Complex Type
<AggregateTarget>	EMModel 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.	Complex Type
<Name>RAC30</Name>	Name of the RAC database.	String (length <=8)
<Type>rac_database</Type>	Aggregate target type.	String (enumeration: "cluster" or "rac_database")
<Storage>	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.	Complex Type
<Type>ASM</Type>	Type of storage.	String (enumeration: "CFS" or "ASM." "RAW" is not supported.)
<Property/>	Properties for storage. See the corresponding table in this section for the storage properties.	Complex Type

Table 4–1 (Cont.) setModel Request Elements

Path Type	Description	Data Type
</Storage>	End of Storage.	Complex Type
<Target>	The number of targets in the `rac_database` aggregate target should be the same as the number of database instances in the cluster database. Each target element contains information about a database instance in the RAC database.	Complex Type
<Name>RAC30_RAC301</Name>	Name of the target, which should be in the format of `<rac_name>_<rac_instance_name>`.	String
<Type>oracle_database</Type>	Target type. In the `rac_database` aggregate target, it should be set to "`oracle_database`."	String
<Host>bjx30</Host>	Name of the host.	String
<Credential>	Credential of the host.	Complex Type
<Name>oracle</Name>	User name.	String
<Password>welcome1 </Password>	Password.	String
</Credential>	End of Credential.	Complex Type
<Property/>	Property of the target. See the corresponding table in this section for the properties of the "`oracle_database`" target type.	Complex Type
</Target>	End of Target.	Complex Type
<Property/>	Property of the `rac_database` aggregate target. See the corresponding table in this section for the properties of the "`rac_database`" aggregate target type.	Complex Type
</AggregateTarget>	End of AggregateTarget.	Complex Type
</EMModel>	End of EMModel.	Complex Type

Table 4–2 provides path types, descriptions, and data types for request header elements.

Table 4–2 Request Header Elements

Path Type	Description	Data Type
<RequestID/>	Uniquely identifies the request. This is mainly used by the client. Enterprise Manager currently does not track this ID.	String
<Source/>	Request source, which is the request operating system.	String
<Destination/>	Destination should be Enterprise Manager in this release.	String
<RequestProperty> <Type>Singleton</Type> <Property> <Name>Platform</Name> <Value>Linux</Value> </Property> </RequestProperty>	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.	

Table 4–3 provides target types, properties, descriptions, and data types for Create RAC.

Table 4–3 Create RAC Properties

Target Type	Property	Description	Data Type
<Target> <Type>host</Type> </Target>	CRS_HOME	Oracle Clusterware home directory. This property must be the same for all hosts in the cluster.	String
	ORACLE_HOME_NAME	Oracle Clusterware home name. This is an optional property. The default value is the cluster aggregate target name.	String
	publicNode	Public node name.	String
	privateNode	Private node name.	String
	vipNode	Virtual node name.	String
<Target> <Type>oracle_database</Type> </Target>	ORACLE_HOME	Database home directory. This property must be the same for all database instances in the RAC database.	String
	ORACLE_HOME_NAME	Database home name. This property is optional. The default value is the `rac_database` aggregate target name.	String
	db_username	Database user for setting monitoring credentials. It should always be SYS in this release.	String
	db_password	Password of the database user. It is the password for SYS and SYSTEM in this release.	String
	oms_username	Oracle Management Services host operating system user name.	String
	oms_password	Oracle Management Services host operating system password.	String
<Storage> <Type>ASM</Type> </Storage>	diskGroupName	ASM disk group name.	String
	diskList	Disk device list. Use a comma (,) as a separator.	String (no space allowed)
	diskString	Search paths for ASM disks.	String (no space allowed)
	redundancy	Redundancy level.	String (enumeration: NORMAL, HIGH, or EXTERNAL)
	asmPassword	ASM SYSDBA password.	String

Table 4–3 (Cont.) Create RAC Properties

Target Type	Property	Description	Data Type
<Storage> <Type>CFS</Type> </Storage>	datafileDestination	Data file directory.	String
<AggregateTarget> <Type>cluster</Type> </AggregateTarget>	softwareImageName	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 "Oracle Clusterware Clone."	String
	s_ocrpartitionlocation	OCR location. Use the comma (,) as a separator. This property is only for the Linux platform.	String (no space allowed)
	s_votingdisklocation	Voting disk location. Use the comma (,) as a separator. This property is only for the Linux platform.	String (no space allowed)
	RESPONSEFILE_VERSION	Response file version. This property is optional. The default value is 2.2.1.0.0. This property is only for the Windows platform.	String
	sl_OHPartitionsAndSpace_valueFromDlg	This property specifies the split-up of disk partitions for OCR/Vdisk locations. This property is only for the Windows platform. See Section , "Response File Properties for the Windows Platform" for more information.	String
	ret_PrivIntrList	This property specifies the interconnects to use. This property is only for the Windows platform. See Section , "Response File Properties for the Windows Platform" for more information.	String
<AggregateTarget> <Type>rac_database</Type> </AggregateTarget>	templateName	Database template name. This property is optional. The default value is `General_Purpose.dbc`	String (file name without path)
	gdbName	Global database name. This property is optional. The default value is the `rac_database` aggregate target name.	String (length <=8)

Table 4–3 (Cont.) Create RAC Properties

Target Type	Property	Description	Data Type
	sid	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.	String (length <=8)
	characterSet	Character set. See the Database Globalization Support guide for details. This property is optional. The default value is UTF8.	String
	nationalCharacterSet	National character set. See the Database Globalization Support guide for details. This property is optional. The default value is UTF8.	String

	initParams	Raw strings for additional input. For example:
		`nls_territory=japan, nls_language=japanese`
		This property is optional. The default value is:
		`nls_lang=american,nls_territory=american`
			String (no space allowed)

	softwareImageName	Name of the software library image for the database home. This property is optional. The default value is the latest active software library image of type "Oracle Database Clone."	String

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)

Target Type	Property	Description	Data Type
<Storage> <Type>ASM</Type> </Storage>	asmPassword	ASM SYSDBA password.	String
<AggregateTarget> <Type>cluster</Type> </AggregateTarget>	s_ocrpartitionlocation	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.	String (no space allowed)
	s_votingdisklocation	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.	String (no space allowed)

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

Target Type	Property	Description	Data Type
	RESPONSEFILE_VERSION	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.	String
	sl_OHPartitionsAndSpace_valueFromDlg	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.	String
	ret_PrivIntrList	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.	String

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)

Target Type	Property	Description	Data Type
<Target> <Type>host</Type> </Target>	CRS_HOME	Oracle Clusterware home directory. This property must be the same for all hosts in the cluster.	String
	ORACLE_HOME_NAME	Oracle Clusterware home name. This property is optional. The default value is cluster aggregate target name.	String
	publicNode	Public node name.	String
	privateNode	Private node name.	String
	vipNode	Virtual node name.	String
<Target> <Type>oracle_database</Type> </Target>	ORACLE_HOME	Database home directory. This property must be the same for all database instances in the RAC database.	String
	ORACLE_HOME_NAME	Database home name. This property is optional. The default value is `rac_database` aggregate target name.	String
	db_username	Database user name that has a SYSDBA role.	String

Table 4–5 (Cont.) Add Node Properties (New Node)

Target Type	Property	Description	Data Type
	db_password	Password of the user above.	String
	oms_username	Oracle Management Services host operating system user name.	String
	oms_password	Oracle Management Services host operating system password.	String

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)

Target Type	Property	Description	Data Type
<Target> <Type>host</Type> </Target>	publicNode	Public node name.	String
	CRS_HOME	Oracle Clusterware home directory. This property must be the same for all hosts in the cluster.	String
<Target> <Type>oracle_database</Type> </Target>	ORACLE_HOME	Database home directories. This property must be the same for all database instances in the RAC database.	String

Table 4–7 provides target types, properties, descriptions, and data types for Delete Node for remaining nodes.

Table 4–7 Delete Node Properties (Remaining Nodes)

Target Type	Property	Description	Data Type
<Target> <Type>host</Type> </Target>	CRS_HOME (or ORACLE_HOME)	Oracle Clusterware home directory.	String
<Target> <Type>oracle_database</Type> </Target>	ORACLE_HOME	Database home directory.	String
	db_username	Database user name that has a SYSDBA role.	String
	db_password	Password of the user above.	String

Table 4–7 (Cont.) Delete Node Properties (Remaining Nodes)

Target Type	Property	Description	Data Type
	oms_username	Oracle Management Services host operating system user name.	String
	oms_password	Oracle Management Services host operating system password.	String
<AggregateTarget> <Type>rac_database</Type> </AggregateTarget>	oms_delete_all_targets	Removes all targets on the instance host, including the host and agent. This property is optional. The default value is False.	String (enumeration: True or False)

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.

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:

```
<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:

```
<Property>
    <Name>sl_OHPartitionsAndSpace_valueFromDlg</Name>
    <Value>{"3","1","486","0","N/A","1","4","1","486","0","N/A","2"}</Value>
</Property>
```

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:
  - 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:

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

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

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

Table 4–8 (Cont.) General Target Properties

Query Path	Description	Data Type
Property(Name:compliance score)	Compliance score as a real number between 0 and 1 (inclusive) of the Enterprise Manager target instance.	Number
Property(Name:last load time)	Last load time of the data for the target instance as the number of milliseconds since January 1, 1970, 00:00:00 GMT.	Number
Host	Enterprise Manager target instance name (of type host) of the host related to the Enterprise Manager target instance.	String

Table 4–9 provides property names, descriptions, and data types for Oracle Database queryable properties. The target type for the Oracle Database is `oracle_database`.

Table 4–9 Oracle Database Properties

Query Path	Description	Data Type
Property(Name:instance name)	Instance name of the database instance.	String
Property(Name:listener)	Listener Enterprise Manager target instance name (of type `oracle_listener`) of the listener for the database instance.	String
Property(Name:is archiving)	The value is 1 if high availability archiving is on for the Oracle database instance. Otherwise, the value is 0.	Integer
Property(Name:is flashback logging)	The value is 1 if high availability flashback logging is on for the Oracle database instance. Otherwise, the value is 0.	Integer

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 4–10 Oracle Listener Properties

Query Path	Description	Data Type
Property(Name:alias)	Alias of the Oracle Listener instance.	String
Property(Name:net address)	Net address of the Oracle Listener instance.	URI
Property(Name:listener.ora location)	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.	String
Property(Name:start name)	Start time of the Oracle Listener instance. The form of this time stamp is not further specified here.	Time

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 4–11 Host Target Properties

Query Path	Description	Data Type
Property(Name:cluster)	Enterprise Manager target instance name (of type cluster) of the cluster for this host instance.	String
Property(Name:cpu utilization)	CPU utilization as a real number between 0 and 1 (inclusive) of the host.	Number
Property(Name:memory utilization)	Memory utilization as a real number between 0 and 1 (inclusive) of the host.	Number
Property(Name:total io rate)	Total I/O per second.	Number

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

Table 4–12 Cluster Properties

Query Path	Description	Data Type
Property(Name:version)	Clusterware version. Note that this property definition just redefines the same property defined for the general target mappings.	String
Property(Name:cluster database)	Cluster databases (of type `rac_database`).	String
Target	Cluster hosts (of type host).	String

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 4–13 Cluster Database Properties

Query Path	Description	Data Type
Property(Name:cluster)	Enterprise Manager target instance name (of type cluster) of the cluster for this database instance.	String
Property(Name:database name)	Database instance name.	String
Property(Name:is archiving)	Value is 1 if high availability archiving is on for the cluster database. Otherwise, the value is 0.	Integer
Target	Cluster database instance of type `oracle_database`.	String

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 4–14 Oracle Enterprise Manager Agent Properties

Query Path	Description	Data Type
Property(Name:management service)	OMS that the Enterprise Manager Agent instance uploads to.	String

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 4–15 Oracle Enterprise Manager Repository Target Properties

Query Path	Description	Data Type
Property(Name:agent count)	Number of Agents for this repository instance.	Integer
Property(Name:target count)	Number of targets for this repository instance.	Integer
Property(Name: administrator count)	Number of administrators for this repository instance.	Integer
Property(Name:session count)	Number of active Oracle management services repository sessions for this repository instance.	Integer
Property(Name:Integer)	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 "name" and "value" sub-properties.	String
Property(Name:tablespace)	Expands to the tablespaces used in the database for this repository instance.	String
Property(Name:oms)	OMSs for this Enterprise Manager repository. This property is expanded into complex property elements in the response as described in Table 4–17. They are keyed by the "name" sub-properties.	String

Table 4–16 provides property names, descriptions, and data types for job properties.

Table 4–16 Job Properties

Query Path	Description	Data Type
JobStatus	Integer status (see Table 4–20) of the most recent execution of the job.	Integer
Property(Name:output)	Last 1024 characters of the job output for the last step of the most recent job execution.	String

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 4–17 Oracle Management Service (OMS) Complex Property

Query Path	Description	Data Type
ComplexProperty(Type:oms). Property(Name:name)	OMS name.	String
ComplexProperty(Type:oms). Property(Name:status)	OMS service status (1 for up or 0 for down) for the OMS given by the peer "name" property.	Integer
ComplexProperty(Type:oms). Property(Name:last error)	Time of the last OMS error as the number of milliseconds since January 1, 1970, 00:00:00 GMT, for the OMS given by the peer "name" property.	Number

Table 4–17 (Cont.) Oracle Management Service (OMS) Complex Property

Query Path	Description	Data Type
ComplexProperty(Type:oms). Property(Name:files pending load)	Number of files pending loading into the OMS for the OMS given by the peer "name" property.	Integer
ComplexProperty(Type:oms). Property(Name:load directory)	Load directory of the OMS for the OMS given by the peer "name" property. The form of the directory path (path separator) is not specified further here.	String
ComplexProperty(Type:oms). Property(Name:oldest load file)	Oldest file to load (in minutes) of the OMS for the OMS given by the peer "name" property.	Number

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

Table 4–18 Database Instance Complex Property

Response Path	Description	Data Type
ComplexProperty(Type:database). Property(Name:name)	Oracle database instance name.	String
ComplexProperty(Type:database). Property(Name:type)	Oracle database instance type (one of `oracle_database` or `rac_database`).	String

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 4–19 Enterprise Manager Status Codes

Status Code	Description
0	Target down
1	Target up
2	Metric error
3	Agent down
4	Unreachable
5	Blackout
6	Pending/unknown

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 4–20 Job Status Codes

Status Code	Description
1	SCHEDULED — The execution is in a scheduled state.
2	RUNNING — The execution is running.
3	INITIALIZATION ERROR — The execution encountered an error and the remote process did not run.
4	FAILED — The execution failed.
5	SUCCEEDED — The execution succeeded.
6	SUSPENDED BY USER — A user suspended the execution.
7	SUSPENDED ON AGENT UNREACHABLE — The execution was suspended because the Agent was unreachable.
8	STOPPED — A user stopped the execution.
9	SUSPENDED ON LOCK — The execution is waiting for a lock on a shared resource.
10	SUSPENDED ON EVENT — The execution is waiting for an event to occur (usually for an Agent to bounce).
11	SUSPENDED ON BLACKOUT — The execution is suspended on a blackout.
12	STOP PENDING — The execution is in Stop Pending status waiting for some running steps to finish.
13	SUSPEND PENDING — The execution is in Suspend Pending status waiting for some running steps to finish.
14	Inactive (internal state).
15	Queued (internal state).
16	Failed retried (internal state).
18	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.
20	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.
21	SUSPENDED ON MISSING CREDENTIALS — The execution is suspended because some of the credentials needed for the job are not set.

5

Error Messages and Debugging

This chapter provides all Management Connector specific error messages and debugging information. The errors are returned in the response model.

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.

Debugging

The Connector Framework uses the log4j logging utility to log the types of messages shown in Table 5–1:

Table 5–1 Message Types and Corresponding Code Names

Message Type	Code Option
Warning	WARN
Error	ERROR
Debugging	DEBUG
Information	INFO

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.

Viewing Debug Messages

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

```
$INSTANCE_HOME/sysman/log/emoms.trc
```

Index

A

C

D

E

G

H

I

J

K

M

N

O

P

Q

R

S

T

U

W

X